EP 26984 (S)

(19) BUNDESREPUBLIK DEUTSCHLAND

DEUTSCHES PATENTAMT

(12) **Offenlegungsschrift**

(10) **DE 41 33 460 A1**

(51) Int. Cl.[5]:
G 06 F 15/66
// H04N 1/41

(21) Aktenzeichen: P 41 33 460.4
(22) Anmeldetag: 9. 10. 91
(43) Offenlegungstag: 15. 4. 93

(30) Unionspriorität: (32) (33) (31)

09.08.91 US 743517

(71) Anmelder:

Ricoh Co., Ltd., Tokio/Tokyo, JP

(74) Vertreter:

Schwabe, H., Dipl.-Ing.; Sandmair, K., Dipl.-Chem. Dr.jur. Dr.rer.nat.; Marx, L., Dipl.-Phys. Dr.rer.nat., Pat.-Anwälte, 8000 München

(72) Erfinder:

Blonstein, Steven M., San Jose, Calif., US; Allen, James D., Santa Cruz, Calif., US; Boliek, Martin P., Palo Alto, Calif., US

Prüfungsantrag gem. § 44 PatG ist gestellt

(54) Einrichtung und Verfahren zum Verdichten von Bildern

(57) Transformationen, beispielsweise DCT, werden bei einer Bildkompression oder -verdichtung verwendet. Eine Transformation, welche der DCT verhältnismäßig nahe kommt, wird für deren arithmetische Vereinfachung bevorzugt. Es ist ein Verfahren und eine Einrichtung beschrieben, mit deren Hilfe die Bildverdichtung ohne Multiplikationen erfolgt, obwohl sie noch mit einer JPEG (Joint Photographic Experts Group) Transformation kompatibel ist. Um die Bildqualität zu verbessern, sind noch weitere Verbesserungen aufgezeigt.

DE 41 33 460 A1

Beschreibung

Die Erfindung betrifft eine Einrichtung und ein Verfahren zum Verdichten von Bildern und betrifft insbesondere eine Einrichtung und ein Verfahren zum Verdichten von Steh- bzw. Einzelbildern, was mit einem JPEG (Joint Photographic Experts Group) Steh- oder Einzelbild-Verdichtungsstandard kompatibel ist.

Wenn hochqualitative Bilder verdichtet oder komprimiert werden müssen, um Speicherplatz zu sparen oder um Übertragungsanforderungen zu genügen, ist es in der Praxis allgemein üblich, zuerst die Bilder an einen anderen Platz zu übertragen, wo die Information kompakter dargestellt werden kann. Hierzu wird üblicherweise Block für Block eine lineare Transformation (eine Matrix-Multiplikation) vorgenommen; eine typische Anordnung besteht darin, 8-Punkt-Transformationen an Zeilensegmenten von 8 Bildelementen durchzuführen und dann 8-Punkt-Transformationen an den 8 Element-Spaltensegmenten dieses zeilen-transformierten Bildes durchzuführen. Genauso kann eine einzige 64-Punkt-Transformation an einem Bildelementblock aus 64 Bildelementen (Pixels) durchgeführt werden, welche in einem (8 × 8) Block angeordnet sind.

Eine gute Wahl für eine eindimensionale Transformation ist die diskrete Chebychev-Transformation:

$$F(u) = c(u) \cdot \sum_{i=0}^{7} f(i) \cdot \cos u(2i+1)\pi/16$$

wobei

$$c(u) = \{1/\sqrt{2}/8 \text{ für } u = 0\}$$
$$\{\text{sonst } 2/8\}$$

Bei dieser Transformation ergeben sich mehrere Vorteile, nämlich a) die Verdichtung bzw. Kompression ist für einige Maßnahmen beinahe optimal; b) es gibt schnelle Rechenalgorithmen, um diese Transformation und deren Umkehr durchzuführen, und c) es kann ein Aufbereiten (deblurring) (ein Verbessern des Ausgangsbildes) ohne weiteres an dem Transformationsraum durchgeführt werden, wozu bereits bestimmte Voraussetzungen gegeben sind, die beispielsweise von Acheroy, M., "Use of the DCT for the restoration of an image sequence", SPIE Vol. 593, Medical Image Processing (1985), beschrieben worden sind.

Gemäß der Erfindung sollen daher eine Einrichtung und ein Verfahren zum Verdichten von Bildern geschaffen werden, welche auch noch mit einer JPEG-Information kompatibel sind. Gemäß der Erfindung ist dies bei einer Einrichtung zum Verdichten von Bildern durch die Merkmale im kennzeichnenden Teil des Anspruchs 1 erreicht. Ferner ist dies bei einem Verfahren zum Verdichten von Bildern durch die Merkmale im kennzeichnenden Teil des Anspruchs 2 erreicht. Vorteilhafte Weiterbildungen sind Gegenstand der Unteransprüche.

Nachstehend wird die Erfindung anhand von bevorzugten Ausführungsformen unter Bezugnahme auf die anliegenden Zeichnungen im einzelnen erläutert. Es zeigt

Fig. 1A ein Blockdiagramm einer Verdichtungs- oder Kompressionseinrichtung gemäß der Erfindung;

Fig. 1B eine Dekompressionseinrichtung gemäß der Erfindung;

Fig. 2A bis 2C eine Eingangsbildelement-Anordnung, eine zeitlich richtige Block- bzw. Vektoreinteilung von Daten gemäß der Erfindung;

Fig. 3A und 3B eine Dreipunkt-Transformation von RGB- in XYZ-Daten;

Fig. 4A und 4B mögliche VLSI-Anordnungen gemäß der Erfindung;

Fig. 5 ein Diagramm eines bei der Erfindung verwendeten Schieberegisters;

Fig. 6A ein Diagramm eines Schiebe-Arrays gemäß der Erfindung;

Fig. 6B ein Beispiel des Schiebe-Arrays der **Fig.** 6A;

Fig. 7 ein Diagramm eines verknüpften Datenflusses;

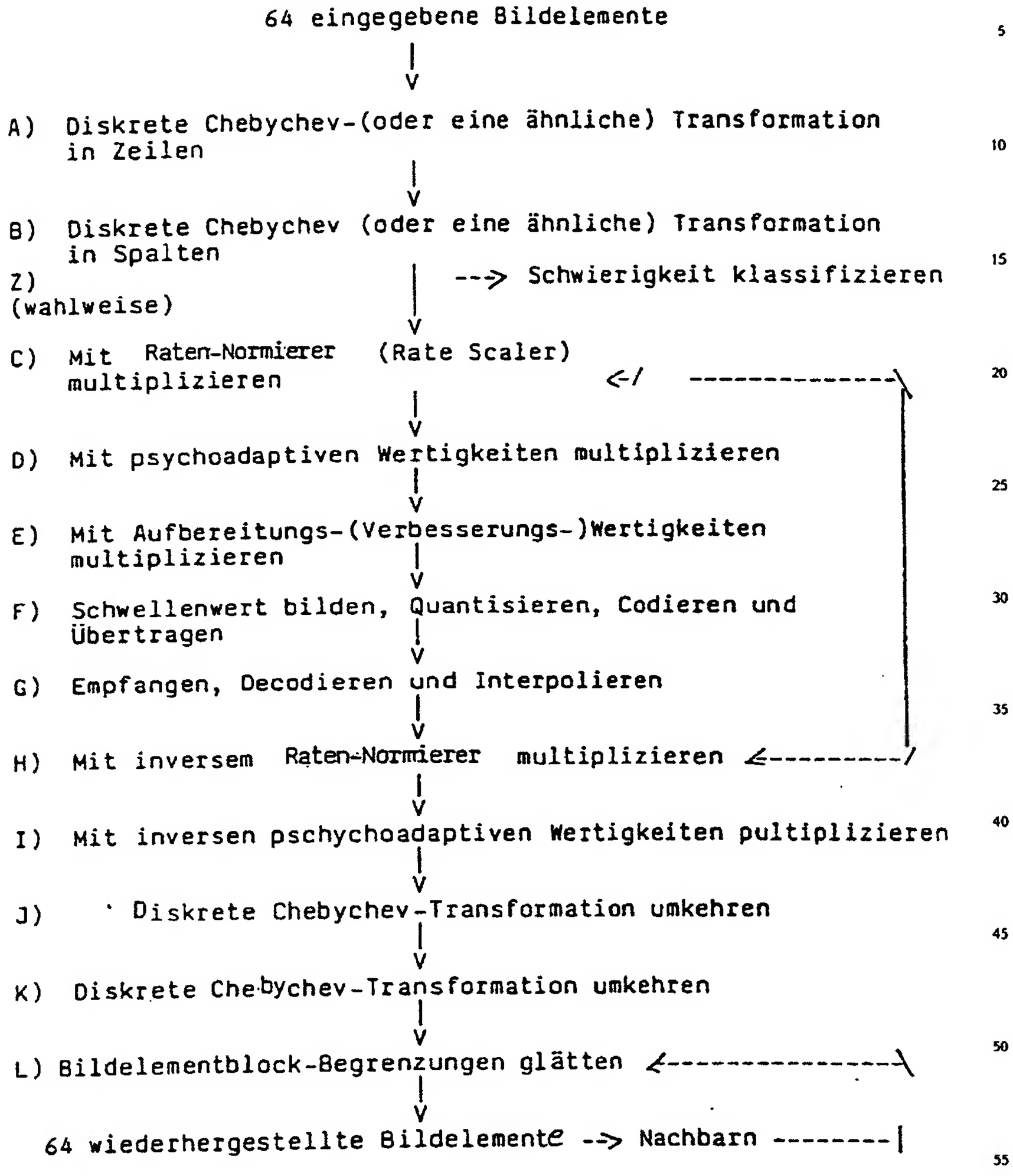
Fig. 8A und 8B Diagramme von Vorwärts-Addier-Arrays gemäß der Erfindung;

Fig. 9 ein Diagramm einer zweidimensionalen verallgemeinerten Chen-Transformation gemäß der Erfindung und

Fig. 10 ein Blockdiagramm einer bevorzugten Ausführungsform der Erfindung.

Ein vollständiges System für die Verdichtung und Wiederherstellung von Bildern kann wie in der folgenden Tabelle 1 aussehen.

Tabelle 1



Die vorstehende Tabelle beschreibt die Erfindung und, wenn die freigestellten Schritte (L, Z) weggelassen sind, ebenso die augenblickliche Technologie. Die Multiplikation mit den Aufbereitungswertigkeiten (Schritt E) kann auch als ein Decodierschritt (beispielsweise nach dem Schritt I) durchgeführt werden.

Das Aufbereiten (Verbessern) ist vorgesehen, um die Punkt-Spreiz-(point-spread-)Funktion der Eingabeeinrichtung auszugleichen. Dies muß an der Einrichtung abgestimmt oder weggelassen werden, wenn das eingegebene Bild bereits verbessert worden ist. Es gibt auch noch andere Wege, das Bild aufzubereiten; die hier aufgezeigte Methode ist rechnerisch preiswert und paßt zu einigen Anwendungsfällen, wie beispielsweise zu einem Farbkopierer.

Die Berechnung der Vorwärts-Transformation (A, B) kann so arrangiert werden, daß viel von der Rechenauslastung aus einer Endmultiplizierstufe besteht. Durch Vorausberechnen der Produkte dieser Multiplikationen

und derjenigen bei den Schritten (CE) kann der Verdichtungsprozeß beschleunigt werden. In ähnlicher Weise kann die Berechnung der umgekehrten Transformation (J, R) arrangiert werden, so daß viel von der Rechenausnutzung aus der vorbereitenden Multiplizierstufe besteht. Wiederum wird durch Vorausberechnen der Produkte die Rechenbeanspruchung von Stufen (H, I) wirksam beseitigt. Außerdem wird eine andere Transformation durch die 2-D-DCT-Transformation ersetzt, wodurch sich eine weitere Rechenvereinfachung ergibt.

Ferner können die psychoadaptiven Wertungen wahlweise geändert werden, um die verknüpften Multiplikatoren für Schritte (B,D) rechnerisch wirksam zu machen, beispielsweise Potenzen von zwei. Kleine Änderungen in den psychoadaptiven Wertungen der niederenergetischen Ausgangstransformationselemente haben eine geringere Wirkung auf die Bildqualität oder Verdichtungsrate.

Schließlich wird die Aufmerksamkeit auf die Schritte (L, Z) in Fig. 1, nämlich auf die Klassifizierung der Bildschwierigkeit und das Glätten von Blockgrenzen gelenkt. Da diese Möglichkeiten freigestellt und unabhängig von dem Grundprinzip der Erfindung sind, wird auf sie in der vorliegenden Beschreibung nur in sehr geringem Umfang eingegangen.

Chen-Algorithmus

Der eindimensionale Chen-Algorithmus (siehe Chen, W., et al., "A fast Computational algorithm for the DCT", IEEE Trans. Commun, Vol. COM-25 [1977]) drückt aus, daß

$$X = s/N A_N x \quad (1)$$

wobei x ein Datenvektor ist, X ein transformierter Vektor und A_N das folgende bedeutet:

$$A_N = c(k)\cos((2j+1)k\pi/2N);$$

$$j,k = 0,1,2,\ldots,N-1.$$

A_N kann auf folgende Weise zerlegt werden

$$A_N = Z_N \left| \begin{array}{cc} A_{N/2} & 0 \\ 0 & R_{N/2} \end{array} \right| B_N \quad (2)$$

wobei $R_{N/2}$ das folgende bedeutet:

$$R_{N/2} = c(2k+1)\cos(2j+1)(2k+1)\pi/2N;$$

$$j,k = 0,1,2,\ldots,N/2-1. \quad (3)$$

Hierbei ist die Matrix Z die Chen-Matrix P. Die Schreibweise ist geändert worden, um in der vorliegenden Anmeldung eine Verwechslung hinsichtlich der Matrix P zu vermeiden.

Achtpunkt-(N = 8)-1D-Chen-Transformationsbeispiel

Für einen Achtpunkt-1D-Chen-Algorithmus wird Gl. (2) recursiv zweimal verwendet. Bei der ersten Iteration werden die Matrizen Z_8, R_4 und R_8 verwendet. Bei der zweiten Iteration wird nach A_4 aufgelöst, und es werden die Matrizen Z_4, R_2, A_2 und B_4 verwendet. Dies kann ohne weiteres aus den vorstehenden Gleichungen oder aus den vorstehenden Veröffentlichungen von Chen et al. abgeleitet werden.

$$A_8 = Z_8 \left| \begin{array}{c|cc|cc|c} & A_2 & 0 & & & \\ Z_4 & & & B_4 & 0 & B_8 \\ & 0 & R_2 & & & \\ \hline & & 0 & & R_4 & \end{array} \right|$$

wobei sind:

$$Z_8 = \begin{vmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{vmatrix}$$

$$B_8 = \begin{vmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{vmatrix}$$

$$R_4 = \begin{vmatrix} C1 & C3 & C5 & C7 \\ C3 & -C7 & -C1 & -C5 \\ C5 & -C1 & C7 & C3 \\ C7 & -C5 & C3 & -C1 \end{vmatrix} = \begin{vmatrix} C1 & C4\,(C7 + C1) & C4\,(C1 - C7) & C7 \\ C3 & C4\,(C5 - C3) & -C4\,(C3 + C5) & -C5 \\ C5 & -C4\,(C5 + C3) & C4\,(C3 - C5) & C3 \\ C7 & C4\,(C7 - C1) & C4\,(C1 + C7) & -C1 \end{vmatrix}$$

$$= \begin{vmatrix} C1 & 0 & C7 & 0 \\ 0 & C3 & 0 & C5 \\ 0 & C5 & 0 & -C3 \\ C7 & 0 & -C1 & 0 \end{vmatrix} \cdot \begin{vmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{vmatrix} \cdot \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & C4 & 0 & 0 \\ 0 & 0 & C4 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \cdot \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{vmatrix}$$

$$= RA_4\, RB_4\, RC_4\, RD_4$$

$$Z_4 = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

$$B_4 = \begin{vmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & -1 & 0 \\ 1 & 0 & 0 & -1 \end{vmatrix}$$

$$R_2 = \begin{vmatrix} C2 & C6 \\ C6 & -C2 \end{vmatrix}$$

$$A_2 = \begin{vmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{vmatrix}$$

wobei aus Gl. 3 sich ergibt:

$Cn = \cos(n\pi/16)$.

Chen-Wu (modifizierte) oder parametrisierte Transformation

Bisher ist alles, was getan worden ist, die Chen-Transformation. Man könnte sie ausmultiplizieren und um unvernünftigerweise eine Rechenersparnis zu realisieren, eine intensive DCT-Implementierung multiplizieren. Dies ist jedoch nicht das, was die Anmelderin vorgesehen hat. Um die Anzahl an Multiplikationen auf das Minimum zu reduzieren, werden die Matrizen folgendermaßen reparametrisiert. Dies ist es, was die Anmelderin die (modifizierte) Chen-Wu-Transformation nennt, welche von ihr geschaffen worden ist.

$$R_4 = \begin{vmatrix} a & r(1+a) & r(a-1) & 1 \\ c & r(1-c) & -r(1+c) & -1 \\ 1 & -r(1+c) & r(c-1) & c \\ 1 & r(1-a) & r(a+1) & -a \end{vmatrix} \begin{vmatrix} 1/\sqrt{(a^2+1)} & 0 & 0 & 0 \\ 0 & 1/\sqrt{(c^2+1)} & 0 & 0 \\ 0 & 0 & 1/\sqrt{(c^2+1)} & 0 \\ 0 & 0 & 0 & 1/\sqrt{(a^2+1)} \end{vmatrix}$$

$$= RE_4\, RF_4$$

$$R_2 = 1/\sqrt{(b^2+1)} \begin{vmatrix} b & 1 \\ 1 & -b \end{vmatrix}$$

$$A_2 = 1/\sqrt{2} \begin{vmatrix} 1 & 1 \\ 1 & -1 \end{vmatrix}$$

wobei gilt:

$a = C1/C7 = \cos(\pi/16)/\cos(7\pi/16) = \tan(2\pi/16)$,
$b = C2/C6 = \tan(6\pi/16)$,
$c = C3/C5 = \tan(5\pi/16)$,
$r = C4 = \cos(4\pi/16)$.

Die Diagonalmatrix RF_4 enthält die Normalisierungsfaktoren der unparametrisierten Matrix RA_4. Ebenso kann eine Diagonalmatrix auch aus den Konstanten in R_2 und A_2 bestehen.

Bei einer Wiederherstellung der A_8-Matrix werden eindeutig zwei Matrizen erhalten. Die Diagonalmatrizen werden gesondert von der Hauptmatrix erhalten. Die Hauptmatrix wird mit den B_N-Termen multipliziert. Nach der entsprechenden Umordnung und Multiplikation mit dem konstanten Term wird Gl. (1) auf das folgende reduziert:

$$X = Q(a, b, c)\, P(a, b, c, r)\, x$$

wobei gilt:

$$Q_{(a,b,c)} = \begin{vmatrix} 1/2\sqrt{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/2\sqrt{(a^2+1)} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/2\sqrt{(b^2+1)} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/2\sqrt{(c^2+1)} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/2\sqrt{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1/2\sqrt{(c^2+1)} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1/2\sqrt{(b^2+1)} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/2\sqrt{(a^2+1)} \end{vmatrix} \quad (5)$$

$$P_{(a,b,c,r)} = \begin{vmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ a & r(a+1) & r(a-1) & 1 & -1 & r(1-a) & -r(a+1) & -1 \\ b & 1 & -1 & -b & -b & -1 & 1 & b \\ c & r(1+c) & -r(c+1) & -1 & 1 & r(c+1) & r(c-1) & -c \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -r(c+1) & r(c-1) & c & -c & r(1-c) & r(c+1) & -1 \\ 1 & -b & b & -1 & -1 & b & -b & 1 \\ 1 & r(1-a) & r(a+1) & -a & a & -r(a+1) & r(1-a) & -1 \end{vmatrix} \quad (6)$$

Verallgemeinerte Transformation

Die verallgemeinerte 8-Punkt-DCT-Transformation ist durch vier Parameter a, b, c, und r bestimmt, und kann geschrieben werden als

$$T_{(a,b,c,r)} = P_{(a,b,c,r)} \times Q_{(a,b,c)}$$

wobei P(), Q() wie oben angegeben sind.

Die Bildtransformation erfordert zwei solcher Transformationen T, nämlich T_v und T_h, um das Bild vertikal bzw. horizontal zu transformieren. Die vollständige zweidimensionale Transformation wird dann dargestellt durch

$$[F] = [T_v][f]^t[T_h]$$

wobei f der eingegebene Bildblock ist, F die ausgegebenen Transformations-Koeffizienten sind und der tiefstehende Buchstabe (t) ein Matrix-Transponieren bezeichnet. Hierbei sind alle Matrizen 8 mal 8.

Da eine Diagonalmatrix (wie Q) ihre eigene Transponierte ist, und

$$[A]^t[B]^t = ([E]^t[A])$$

für alle Matrizen und

$$[T_v] = [F_v][Q\,1_v].$$

$$[T_h1] = [P_h][Q_h]$$

gilt, kann geschrieben werden

$$[F] = [Q_v][P_v][f][P_h][Q_h]$$

was sich reduziert auf

$$F(i,j) = q(i,j) \cdot g(i,j)$$

wobei gilt

$$[g] = [F_v][f][P_h^t]$$

und

$$q(i,j) = Q_v(i,i) \cdot Q_h(j,j) \quad (7)$$

Zum Transformieren eines Bildblockes wird [g] mit Hilfe des Chen-Wu-Algorithmus aufgelöst und dann mit den Faktoren q (i, j) multipliziert. Wenn gegeben ist

$$P_v = P(a, b, c, r_v)$$

und

$$P_h = P(a, b, c, r_h)$$

wird die Umkehr der vorstehenden Transformation ausgedrückt durch

$$[f] = [P'_v][Q_v][F][Q_h][P'^t_h]$$

wobei ist:

$$P'_v = P(a, b, c, 1/2\, r_v)$$

und

$$P'_h = P(a, b, c, 1/2\, r_h).$$

Wieder gibt es eine Lösung über den Chen-Wu-Algorithmus.

Chen-Algorithmus

Es sind mehrere Methoden gefunden worden, um die Berechnung der 1-D- oder 2-D-Chebychev-Transformation und deren Umkehr zu beschleunigen. Es gibt einen bekannten Algorithmus (siehe Chen, W., et al. wie oben angegeben und Cooley and Tukey, JW, "An algorithm for (fast Fourier series", _Math Comput_, XIX, Nr. 90, Stn 296—301, 1965), bei welchem ein beliebiger 8-Tuple mit der vorstehend wiedergegebenen Matrix T multipliziert wird, wobei nur 16 Multiplikationen, 13 Additionen und 13 Subtraktionen angewendet werden. Dieser Algorithmus ist nicht auf irgendwelche speziellen Verhältnisse der Parameter a, b, c und r angewiesen.

(Modifizierter) Chen-Wu-Algorithmus

Durch Zerlegen von [T]=[P][Q], wie oben beschrieben, in Faktoren wird der Chen-Algorithmus in zwei Stufen aufgeteilt, wobei 8 Multiplikationen bei der Multiplikation [Q], 8 Multiplikationen und der Rest der Arithmetik bei der Multiplikation mit [P] verwendet werden. Dies ist eine Folge der Wahl für [Q]; mehrere Elemente von [P] werden 1 oder —1, und eine Multiplikation erübrigt sich.

Wie oben aufgezeigt, werden ähnliche Vereinfachungen bei der inversen Transformation, der 2-D-Transformation und der inversen 2-D-Transformation angewendet. Für 8 mal 8 Blöcke werden 128 Multiplikationen entwender für die Vorwärts- oder die Umkehr-2-D-Transformation (ausgenommen die Multiplikationen mit [q]) verwendet. Wenn der interne Datenfluß des Chen-Algorithmus betrachtet wird, sind diese Multiplikationen in eine Struktur von acht Addier-/Subtrahier-Stufen und vier Multiplikations-Stufen eingebettet.

Es ist wichtig zu betonen, daß der Chen-Algorithmus ohne Rücksicht auf die Parameter a, b, c und r arbeitet. Jedoch hat die bisher angewendete 8-Punkte-Diskrete-Chebychev-Transformation (DCT) die Parameter der "echten Cosinus-Transformation" verwendet:

$$a = \text{tang}(7 \cdot \pi/16)$$
$$b = \text{tang}(6 \cdot \pi/16)$$
$$c = \text{tang}(5 \cdot \pi/116)$$
$$r = \sqrt{(1/2)} = 0{,}70710678\ldots$$

wobei die Wahl von r, welche für die Matrix T notwendig und hinreichend ist, orthogonal ist.

Wahl von Parameterwerten

Die Chen-Transformation arbeitet unabhängig von den Werten, welche für die Parameter a, b, c und r ausgewählt worden sind, und zwar deswegen, da die durch QP geschaffene Transformation orthogonal ist. Es ist auch möglich, irgendwelche Zahlen zu verwenden und eine Tranformation zu erhalten, welche die gewünschte Dekorrelation der für eine Verdichtung notwendigen Bilddaten durchführt. Diese Transformation ist keine Diskrete-Cosinus-Transformation noch ist sie eine Annäherung an eine DCT-Transformation. Es ist vielmehr eine eigene Transformation.

Jedoch ist man sich für eine wirksame Dekorrelation des eingegebenen Bildes und für eine Tranformation in verhältnismäßig sinnvolle Raumfrequenz-Koeffizienten allgemein einig, daß die DCT-Transformation sehr wünschenswert ist (siehe Lee, BC, "A Fast Cosine Transform", IEEE ASSP, Vol. XXXIII, 1985). Folglich werden, um die Vorteile der DCT-Transformation zu erreichen, die Parameter entsprechend gesetzt, um sie denjenigen DCT-Transformationen anzunähern, welche in Gl. (4) wiedergegeben sind. Der entgegengesetzte Faktor ist die

Effizienz einer Berechung. Da eine Addition preiswerter durchzuführen ist als eine Multiplikation, werden die Parameter entsprechend gewählt, um rechnerisch wirksam zu sein.

Alternative Algorithmen

Statt der Diskreten Chebychev-Transformation sind andere rechnerische Lösungen gefunden worden. Beispielsweise führt ein Algorithmus, welcher Lee zuzuschreiben ist, die 8-Punkt-1-D- und die 64-Punkt-2D-Transformationen mit 12 bzw. 144 Multiplikationen durch (siehe Wu, HR and Paolin, FJ, "A 2D Fast Cosine Transform", IEEE Conf. on Image Processing, Vol. I, 1989, und Lee, BC, wie oben angegeben). Jedoch gibt es mehrere Nachteile dieser "schnelleren" Algorithmen im Vergleich mit dem Chen-Algorithmus:

a) Die Vereinfachung $T = P \times Q$ (und die entsprechende Zerlegung in Faktoren für die umgekehrte Transformation) ist nicht länger wirksam.
Ein Teilen der diagonalen Matrix Q ist wesentlich für die Vereinfachung.
b) Diese Algorithmen funktionieren nicht mit beliebigen Parametern a, b, c und r. Statt dessen verlassen sie sich auf verschiedene trigonometrische Identitäten, die insbesondere für die wahren Cosinusparameter gültig sind.
c) Diese Algorithmen haben eine schwierige Struktur.
Dies kann die Ingenieurarbeit behindern und die Möglichkeit für eine numerische Instabilität vergrößern.

Erörterung der Erfindung

A) Unter Hinweis auf Tabelle 1 ist zu bemerken, daß die Schritte (C, D, E) in die aus [Q] abgeleiteten Vorwärtstransformation-Nachmultiplikatoren gefaltet werden können. Ebenso können die Schritte (H, I) in die inversen Transformationen-Vormultiplikatoren gefaltet werden, und zwar deswegen, das die Raten-Scalar-Operation, die psychoadaptive Bewertungsoperation (die im allgemeinen als Quantisierungswerte bekannt sind), und die Aufbereitungsbewertungsoperationen alles Punkt-Multiplikationsoperationen sind. Wenn b, c, d, e die Ausgangswerte der Schritte B, C, D bzw. E sind, dann gilt

$$c(i,j) = b(i,j) \cdot q(i,j)$$
$$d(i,j) = c(i,j) \cdot r(i,j) = b(i,j) \cdot q(i,j) \cdot r(i,j)$$
$$e(i,j) = d(i,j) \cdot u(i,j) = b(i,j) \cdot q(i,j) \cdot r(i,j) \cdot u(i,j)$$

oder

$$e(i,j) = b(i,j) \cdot all(i,j)$$

wobei ist:

$$all(i,j) = q(i,j) \cdot r(i,j) \cdot u(i,j) \quad (8)$$

und q (i, j) der Ratenscalar ist, r (i, j) psychoadaptiv ausgewählte (oder sogar vom Benutzer ausgewählte) Quantisierungsbewertungen sind, und u (i, j) Aufbereitungs-(deblurring-)Bewertungen sind. In ähnlicher Weise können Schritte H und I verknüpft werden.

Dies bedeutet in Wirklichkeit, daß die Ratenvormierung adaptive Bewertungs- und Aufbereitungsfunktionen mit keinem zusätzlichen Rechen-Overhead versehen sind. Wie vorstehend bereits erwähnt, ist diese Lösung bei den "schnelleren" Algorithmen wie dem von Lee, nicht anwendbar.

B) Da der Chen-Algorithmus mit irgendwelchen Parametern a, b, c und r arbeitet, können Werte gewählt werden, welche eine Qualität und Verdichtung ähnlich der DCT-Transformation bieten, welche aber zu einer hochschnellen Multiplikation führen. Die folgenden Parameter sind ziemlich nahe bei denjenigen der DCT-Transformation gewählt, aber rechnerisch noch viel wirksamer: $a = 5{,}0$; $b = 2{,}5$; $c = 1{,}5$ und $r = 0{,}75$. Eine Multiplikation ist nunmehr durch eine viel einfachere Arithmetik ersetzt. Ein Multiplizieren mit 5 wird beispielsweise Kopieren; Verschieben nach links-2; Addieren. Ein Multiplizieren mit 1,5 wird Kopieren; Verschieben nach rechts-1; Addieren.

Umgekehrt kann der inverse Zähler eines rationalen Multiplikators in den kombinierten Multiplikator [q] zerlegt werden. Folglich kann das Multiplizieren mit 2,5 ein Multiplizieren mit 5 und 2 für beeinflußte bzw. nicht-beeinflußte Terme werden.

Bei dieser zuletzt erwähnten Überlegung erfordert ein Handhaben des Parameters $r = 0{,}75$ in dem unkomplizierten Chen-Algorithmus 96 Multiplikationen mit 4 und 32 Multiplikationen mit 3. Mit Hilfe des Wu-Paolini-Algorithmus in einer 2D-Implementierungs-Verbesserung ist die gesamte Multiplikationsstufe entfallen, und dies werden dann 36 Multiplikationen mit 16, 24 Multiplikationen mit 12 und 4 Multiplikationen mit 9 (bei der inversen Transformation werden 36 Multiplikationen mit 9, 24 Multiplikationen mit 6 und 4 Multiplikationen mit 4 verwendet).

Auf Kosten der Rechengeschwindigkeit können Parameterwerte, die sogar näher der Cosinus-Transformation sind, gewählt werden. Die Substitutionen $b = 12/5$ und/oder $r = 17/24$ sind möglich. Eine weitere interessante Alternative ist:

$$r_{Zeile} = 0{,}708333 \quad (17/24)$$

$r_{Spalte} = 0,7$ (7/10)

Hier werden leicht unterschiedliche Transformierte (verschiedene Parameter r) für die Zeilen und Spalten verwendet. Dies wird getan, um die Multiplikationen zu vereinfachen, die bei der Wu-Paolini-Methode abgeleitet worden sind. Hier ergibt dieses Verfahren 36 Multiplikationen mit 15, 12 Multiplikationen mit 85/8, 12 Multiplikationen mit 21/2 und 4 Multiplikationen mit 119/16. (Bei der inversen Transformation werden 36 Multiplikationen mit 119/16, 12 Multiplikationen mit 85/16, 12 Multiplikationen mit 24/4 und 4 Multiplikationen mit 15/4 verwendet.)

Auf die gerade beschriebene Weise werden alle Multiplikationen schnell und preiswert durchgeführt, außer für den kombinierten Multiplikator [q] in dem Verdichter (Kompressor) und für den kombinierten Multiplikator [q] in dem Dekompressor. Diese erfordern jeweils eine Multiplikation pro Transformationselement. Das letztere wird dadurch vereinfacht, daß die Mehrzahl der Transformationskoeffizienten Null wird und die meisten der Nicht-Null-Koeffizienten ganze Zahlen ziemlich nahe bei Null sind, welche speziell gehandhabt werden können.

C) Eine weitere Methode wird angewendet, um die Rechenkosten des kombinierten [q]-Multiplikators in dem Kompressor zu reduzieren. Da der "Ratenscaler" tatsächlich ein beliebiger Wert ist, wird er Punkt für Punkt eingestellt, um allen [q]-Matrix-Elementen rechnerisch einfache Werte zu geben, z. B. Potenzen von 2. Diese 64 Einstellungen müssen nur einmal durchgeführt werden (nachdem die "Ratenscaler"- und Aufbereitungs-(deblurring-)Filter spezifiziert werden).

Wenn beispielsweise ein Element (C) des kombinierten Multiplikators und das entsprechende Dekompressions-Multiplikatorelement (D) sich zu $C = 0,002773$ und $D = 0,009367$ ergeben, kann die Näherung $C\sim = 3/1024 = 1024 = 0,002930$ gefunden und verwendet, um die Multiplikation zu vereinfachen. Dies ergibt $C' = 3/1024$ und $D' = D \cdot C/C'\sim = 0,008866$.

Ausführliche Beschreibung des Haupt-)Verfahrens

Anmerkung:

a) In dem quantisierten Transformationsraum ist es üblich und wirksam, die Nicht-Null-Schritte der "AC"-Koeffizientenquantisierung so zu nehmen, daß sie eine konstante Breite (w) haben und den Null-Schritt so nehmen, daß er eine Breite (w • q) hat.
Darüber ist $q = 2$ arithmetisch üblich und ist beinahe optimal für Qualität über einen breiten Bereich von Kompressionsfaktoren. In der Beschreibung ist $q = 2$ genommen ("Doppel-Breite Null"), obwohl die Erfindung alle möglichen q einschließt.

b) Der folgende Algorithmus ist ausgelegt für eine begrenzt-präzise Zweierkompliment-Binär-Ganzzahl-Arithmetik außer für die Zwischenbestimmungen in den Stufen (2), (4) und (8), welche einmal mit hoher Präzisionsarithmetik ausgeführt werden.
Ferner werden mit der zusätzlichen Ausnahme bei Schritt (9.1) die hier beschriebenen ganzzahligen Multiplikationen hinsichtlich der Kosten und der Geschwindigkeit optimiert. Beispielsweise sind die Multiplikationen zu berücksichtigen durch

$$Nrr \cdot Nrc = Drr' \cdot Drc' = 1,75 \cdot 4,25 = 7,4375$$

Durch Wählen der Indentität $7,4375 = (8 - 1) \cdot (1 = 1/16)$ wird die Multiplikation mit Verschiebungen und Additionen wirksam durchgeführt.

c) Die Aufbereitungsmultiplikationen (deblurring multiplies) sind hier beim Schritt 8 dargestellt, sie sollten jedoch üblicherweise, wenn überhaupt, beim Schritt 4 vorgenommen werden. Bei vielen Anwendungen "weiß" der Dekompressor nicht, wie und ob das Bild aufzubereiten ist. Die besten Werte von Thr() hängen von der Eingabeeinrichtung und dem Aufbereitungsverfahren ab.
Eine empfohlene Näherung muß für den Wert m(i,j) (siehe Schritt 8) in der Kompressionszeit (Schritt 4) berechnet und als Teil des verdichteten Bildes übertragen oder gespeichert werden.

d) Es gibt mehrere Wege, die Berechnungen, welche folgen, parallel bzw. zeitlich sequentiell durchzuführen oder zu verschachteln. Die bevorzugte Methode für eine vorgegebene Hardware-Architektur ist unkompliziert.

Beispiel einer Pseudocode-Ausführungsform

Dieser Teil der Anmeldung ist im wesentlichen eine Ausführungsform der Erfindung, welche in Text und Pseudocode erläutert wird. Es gibt mehrere Abschnitte, einschließlich einer Parametrisierung, einer Berechnung von all(i,j) wie in Gl. (8), eine Ausführung des Hauptteils der Vorwärts-GCT, ein Berechnen der inversen all(i,j), eine Ausführung des Hauptteils der inversen GCT.

1. Die Parameter a, b, c und r sind oben dargestellt. Hierbei gibt es einen r-Wert sowohl für Zeilen und Spalten. Obwohl die 2D-GCT eine abtrennbare Transformation ist und in zwei Schritten durchgeführt werden kann, gibt es keine Beschränkung, die es erfordert, daß sie symmetrisch ist. Daher können die Nummierfaktoren, wie dargestellt, asymmetrisch sein.

mögliche Kombinationen von Zähler und Nenner, welche gleich den vorstehenden Werten sein können. Der Entwerfer der GCT-Realisierung hat einen Rückstand (leeway) in den tatsächlichen Werten, welche in Addiereranordnung verwendet worden sind. Die Wahlmöglichkeiten von Werten werden in der Endmultiplikationsstufe korrigiert.

Zu wählen sind

$\tan 7 \cdot \pi/16$	~ = a	= Na/DA
$\tan 6 \cdot \pi/16$	~ = b	= Nb/Db
$\tan 5 \cdot \pi/16$	~ = c	= Nc/Dc
$\sqrt{(0,5)}$	~ = rZeile	= Nrr/Drr
$\sqrt{(0,5)}$	~ = rSpalte	= Nr/Drc
0,5/rZeile	~ = rZeile'	= Nrr'/Drr'
0,5/rSpalte	~ = rSpalte'	= Nrc'/Drc'

Als die Parameter der verallgemeinerten Chen-Transformation, wie oben ausgeführt. Die "Zähler" N und die "Nenner" D müssen nicht ganzzahlig sein, obwohl sie für ein bequemes Rechnen so gewählt werden. Unter mehreren brauchbaren Möglichkeiten ist:

Na = 5	Da = 1
Nb = 3	Db = 1.25
Nc = 1.5	Dc = 1
Nrr = 1.75	Drr = 2.5
Nrc = 4.25	Drc = 6
Nrr' = 1.25	Drr' = 1.75
Nrc' = 3	Drc' = 4.25

Die Erfindung weist jedoch wieder alle rationalen Näherungen zu den vorstehenden Tangenswerten auf. Hierdurch werden dann die benötigten Normierungsfaktoren berechnet.

2. Auch ist zu schreiben:

$U(0) = U(4) = \sqrt{(0,5)}$
$U(1) = U(7) = 1/\sqrt{(Na \cdot Na + Da \cdot Da)}$
$U(2) = U(6) = 1/\sqrt{(Nb \cdot Nb + Db \cdot Db)}$
$U(3) = U(5) = 1/\sqrt{(Nc \cdot Nc + Dc \cdot Dc)}$

3. i soll ein Index bei {0,1,2,3,4,5,6,7} sein, die eine vertikale Position (in dem Bildraum) oder eine Folge einer vertikalen Änderung (in dem Transformationsraum) anzeigen.
j soll ein Index bei {0,1,2,3,4,5,6,7} sein, welche eine horizontale Position (in dem Bildraum) oder eine Folge einer horizontalen Änderung (in dem Transformationsraum) anzeigen.
Debl(i,j) bezeichnen die Aufbereitungs-(deblurring-)Faktoren oder Debl() = 1, wenn nicht aufbereitet wird.
Thr(i,j) bezeichnet die inversen psychoadaptiven Bewertungen, wie sie beispielsweise von CCITT empfohlen worden sind.
M bezeichnet den "Raten-Normierungsfaktor" (raten-scaler); hier ist M = 1 (ungefähr) für typische Verdichtungsraten.
v(i,j) bezeichnet mehrere Luminanzwerte in dem Bildraum (räumlich).
L(i,j) bezeichnet die transformierten Luminanzwerte in dem Transformations-(verdichteten) Raum.
S ist eine beliebige kleine ganze Zahl, welche die arithmetische Präzision in der Rekonstruktion bezeichnet.

Die psychoadaptiven Wertungen i/Thr(i,j) sollten für jeden Satz Parameter der verallgemeinerten Chen-Transformation reoptimiert werden. Jedoch liegen die Parameter bei dem vorstehend angeführten Schritt (1) hinreichend nahe bei den CCITT-Parametern, so daß dieselbe Matrix Thr() optimal ist.

4. Im vorliegenden Fall ist g(i,j) äquivalent all(i,j). Es ist über die 64-Transformationsstellen (i,j) zu iterieren, um sie nach k(i,j) und s(i,j) aufzulösen, um folgendem zu genügen:

$$g(i,j) < \frac{2^{s(i,j)} \cdot M \cdot U(i) \cdot U(j)}{k(i,j) \cdot Zr(i) \cdot Zc(j) \cdot Thr(i,j)}$$

wobei die rechte Seite so nahe wie möglich bei g(i,j) liegt, s(i,j) eine ganze Zahl ist und wobei gilt:

$g(i,j) = 1.0$, $k(i,j)$ in (1,3,5,7,9)
für $i + j < 4$

$g(i,j) = 0.9$, $k(i,j)$ in (1,3,5)
für $i + j < 4$

$g(i,j) = 0.7$, $k(i,j) = 1$
für $i + j < 4$

Zr(i) = i, wenn i = 0, 1, 2 oder 3
Zr(i) = Drr, wenn i = 4, 5, 6 oder 7
Zc(j) = 1, wenn j = 0, 1, 2 oder 3
Zc(j) = Drc, wenn j = 4, 5, 6 oder 7
Zr'(i) = 1, wenn i = 0, 1, 2 oder 3
Zr'(i) = Drr', wenn i = 4, 5, 6 oder 7
Zc'(j) = 1, wenn j = 0, 1, 2 oder 3
Zc'(j) = Drc, wenn j = 4, 5, 6 oder 7

Hierbei sollen die Faktoren g(i,j) unabhängig von der Wahlgröße die Quantisierung-Abweichung (bias) bilden.

5. Ausführung der Vorwärts-GCT

Schritt 5 ist die Pseudocode-Ausführung der Vorwärts-Transformation. Bei den folgenden Schritten wird eine 2D-Transformation in einer verschachtelten Form durchgeführt. Hierbei ist das Bild zu iterieren, indem das Folgende an jedem (8-mal-8-)Block von Luminanzwerten v(,) durchgeführt wird:

5.1 Vorbereiten der Werte

M(i,0) = V(i,0) + V(i,7)
M(i,1) = V(i,1) + V(i,6)
M(i,2) = V(i,2) + V(i,5)
M(i,3) = V(i,3) + V(i,4)
M(i,4) = V(i,3) – V(i,4)
M5(i) = V(i,2) – V(i,5)
M6(i) = V(i,1) – V(i,6)
M(i,5) = M6(i) + M5(i)
M(i,6) = M6(i) – M5(i)
M(i,7) = v(i,0) – v(i,7)
für I = 0, 1, 2, . . ., 7

5.2 Vorbereiten der Werte

H(0,j) = M(0,j) + M(7,j)
H(1,j) = M(1,j) + M(6,j)
H(2,j) = M(2,j) + M(5,j)
H(3,j) = M(3,j) + M(4,j)
H(4,j) = M(3,j) – M(4,j)
H5(j) = M(2,j) – M(5,j)
H6(j) = M(1,j) – M(6,j)
H(5,j) = H6(j) + H5(j)
H(6,j) = H6(j) = H5(j)
H(7,j) = M(0,j) – M(7,j)
für j = 0, 1, 2, . . ., 7

5.3 Jedes H(i,j) multiplizieren mit

(wenn i = 0, 2, 3 oder 4:)

Nrc, wenn j = 5 oder 6
Drc, wenn j = 4 oder 7
1 (keine Aktion), wenn j = 0, 1, 2 oder 3

(wenn i = 4 oder 7:)

Drr Nrc, wenn j = 5 oder 6
Drr Drc, wenn j = 4 oder 7
Drr, wenn j = 0, 1, 2 oder 3

(wenn i = 5 oder 6:)
Nrr Nrc, wenn j = 5 oder 6
Nrr Drc, wenn j = 4 oder 7
Nrr, wenn j = 0, 1, 2 oder 3

5.4 Vorbereiten der Werte

E(0,j) = H(0,j) + H(3,j)
E(1,j) = H(7,j) + H(5,j)

$$E(2,j) = H(0,j) - H(3,j)$$
$$E(3,j) = H(7,j) - H(5,j)$$
$$E(4,j) = H(1,j) + H(2,j)$$
$$E(5,j) = H(6,j) - H(4,j)$$
$$E(6,j) = H(1,j) - H(2,j)$$
$$E(7,j) = H(6,j) + H(4,j)$$

$$F(0,j) = E(4,j) + E(0,j)$$
$$F(4,j) = E(0,j) + E(4,j)$$
$$F(2,j) = Db \cdot E(6,j) + Nb \cdot E(2,j)$$
$$F(6,j) = Db \cdot E(2,j) - Nb \cdot E(6,j)$$
$$F(1,j) = Da \cdot E(7,j) + Na \cdot E(1,j)$$
$$F(7,j) = Da \cdot E(1,j) - Na \cdot E(7,j)$$
$$F(3,j) = Dc \cdot E(5,j) + Nc \cdot E(3,j)$$
$$F(5,j) = Dc \cdot E(3,j) - Nc \cdot E(5,j)$$

für $j = 0, 1, 2, \ldots, 7$

5.5 Vorbereiten der Werte

$$Z(i,0) = F(i,0) + F(i,e)$$
$$Z(i,2) = F(i,0) - F(i,3)$$
$$Z(i,4) = F(i,1) + F(i,2)$$
$$Z(i,6) = F(i,1) + F(i,2)$$
$$Z(i,1) = F(i,7) + F(i,5)$$
$$Z(i,3) = F(i,7) - F(i,5)$$
$$Z(i,5) = F(i,6) - F(i,4)$$
$$Z(i,7) = F(i,6) + F(i,4)$$

$$G(i,0) = Z(i,4) + Z(i,0)$$
$$G(i,4) = Z(i,0) - Z(i,4)$$
$$G(i,2) = Db \cdot Z(i,6) + Nb \cdot Z(i,2)$$
$$G(i,6) = Db \cdot Z(i,2) - Nb \cdot Z(i,6)$$
$$G(i,1) = Da \cdot Z(i,7) + Na \cdot Z(i,1)$$
$$G(i,7) = Da \cdot Z(i,1) - Na \cdot Z(i,7)$$
$$G(i,3) = Dc \cdot Z(i,5) + Nc \cdot Z(i,3)$$
$$G(i,5) = Dc \cdot Z(i,3) - Nc \cdot Z(i,5)$$

für $i = 0, 1, 2, \ldots, 7$

Andernfalls kann die Transformation durch eine eindimensionale Transformation in zwei Schritten aufgeteilt werden. Das Folgende ist ein Beispiel eines eindimensionalen Transformationsschrittes. In Fig. 8 sind diese Schritte dargestellt.

$$A1 = X0 + X7$$
$$A2 = X3 + X4$$
$$A3 = X2 + X5$$
$$A4 = X1 + X6$$
$$A5 = X0 - X7$$
$$A6 = X1 - X6$$
$$A7 = X2 - X5$$
$$A8 = X3 - X4$$

$$B1 = A1 - A2$$
$$B2 = A1 + A2$$
$$B3 = A3 + A4$$
$$B4 = A4 - A3$$
$$B5 = A6 + A7$$
$$B6 = A6 - A7$$

$$C1 = 1.25\,B1$$
$$C2 = 3\,B1$$
$$C3 = 1.25\,B4$$
$$C4 = 3\,B4$$
$$C5 = 1.5\,A5$$
$$C6 = 1.0625\,B5$$
$$C7 = 1.0625\,B6$$
$$C8 = 1.5\,A8$$

$$D1 = C5 + C6$$
$$D2 = C5 - C6$$
$$D3 = C7 + C8$$
$$D4 = C7 - C8$$

$$E1 = 2.5\,D1$$
$$E2 = 1.25\,D2$$
$$E3 = 2.5\,D3$$
$$E4 = 1.5\,D4$$

$$Y0 = B2 + B3$$
$$Y1 = E1 + (0.5\,D3)$$
$$Y2 = C2 + C4$$
$$Y3 = E2 + D4$$
$$Y4 = B2 - B3$$
$$Y5 = D2 - E3$$
$$Y6 = C1 - C4$$
$$Y7 = <0.5\,D1) - E4$$

Alle Vielfachen in diesen Gleichungen werden mit Schiebe- und Addieroperationen durchgeführt.

Um dies zu der Matrixform der GCT in Beziehung zu bringen, wird ein Beispiel an dem Vektorpunkt Y6 demonstriert.

$$Y6 = C1 - C4 = (1{,}25\,B1) - (3\,B4) = 1{,}25\,(A1 - A2) - 3\,(A4 - A3)$$

$$= 1{,}25\,((X0 + X7) - (X3 + X4)) - 3\,((X1 + X6) - (X2 + X5))$$

$$= 1{,}25\,X0 - 3\,X1 + 3\,X2 - 1{,}25\,X3 + 1{,}25\,X4 + 3\,X5 - 3\,X6 + 1{,}25\,X7$$

$$Y6/1{,}25 = X0 - 2{,}4\,X1 + 2{,}4\,X2 - X3 + X4 + 2{,}4\,X5 - 2{,}4\,X6 + X7$$

$$= |1 \quad -b \; b \quad -1 \; 1 \quad b \quad -b \; 1| \quad x$$

wobei b = 2,4 ist. Dies ist die sechste Zeile der Matrix P in der Gleichung. Das Teilen durch 1,25 ist ein Normierungsfaktor, welcher in der Ratennormier-(rate scaler)Matrix gesammelt wird.

Die Zeilendaten eines 8 × 8-Bildelementblocks durchlaufen diese Addieranordnung. Die sich ergebenden eindimensionalen Häufigkeitskomponenten werden umgestellt und durchlaufen wieder dieselbe Anordnung.

6. Nach dem Schritt 5.5 in jedem Bild-Unterblock und für jede der 64 Stellen (i,j), wobei k(i,j) und s(i,j) aus Schritte (4) verwendet werden, wird der folgende Wert vorbereitet:

$$L(i,j) = G(i,j) \cdot k(i,j) \cdot 2^{-s(i,j)}$$

wenn aber dies negativ ist (oder i = j = 0), ist 1 dazu zu addieren. Dies Ergebnis ist dann der Transformationskoeffizient L(i,j).

Anmerkung zu Schritt 6:

Die hier durchgeführten Berechnungen sind einfach, da

- k(i,j) immer 1, 3, 5, 7 oder 9 und üblicherweise 1 ist
- eine Multiplikation mit 2 ^ (−s(i,j)) einfach eine Verschiebung nach rechts ist (oder vielleicht eine Verschiebung nach links, wenn M sehr groß gewählt wurde).

Bei Stellenverschiebungen nach rechts wird immer nach unten gerundet. Tatsächlich wird ein Runden gegen Null gewünscht. Daher die Klausel "wenn (negativ) 1 hinzuaddieren". Bei der Addition von 1 wird, wenn i = j = 0 ist, auf v(i,j) ≧ 0 vertraut, und es gibt gerade eine Einrichtung, um die Anweisung von Schritt (9.1) zu vereinfachen.

7. Die Werte L(i,j) sind zu codieren, zu speichern und/oder zu übertragen. Schließlich werden sie wieder gewonnen und das Bild wird durch die folgenden Schritte rekonstruiert.

8. Dies ist die Umkehrversion von all(i,j). Dann ist über die 64 Transformationsstellen (i,j) zu iterieren und nach m(i,j) als der nächstgelegenen ganzen Zahl aufzulösen:

$$m(i,j) = \frac{U(i)^2 \cdot U(j)^2 \cdot Zr(i) \cdot Zc(j) \cdot Debl(i,j)}{4\text{-}S\text{-}s(i,j)} \Big/ Zr'(i) \cdot Zc'(j) \cdot k(i,j) \cdot 2,$$

wobei s(i,j) und k(i,j) bei dem Schritt (4) gelöst werden, und wobei die Ausdrücke "Z" beim Schritt (4) definiert sind.

Ebenso ist A(i,j) als die nächstliegende ganze Zahl zu wählen:

$$A(0,0) = \frac{2^{S-2}}{Drc' \cdot Drr'} - 0.5 \cdot m(0,0)$$

$$A(i,j) = m(i,j) \cdot (25 - i - j)/64$$
für i = 0 oder j = 0

Anmerkungen zu Schritt 8:

Die Werte m(i,j) können beim Schritt (4) vorausberechnet und zusammen mit dem verdichteten Bild übertragen worden sein. Dies ist für A(i,j) nicht notwendig, welches nur von Konstanten und m(i,j) abhängt. Bei Anwendungen, bei welchen der "Ratenormierungsfaktor" (rate scaler) und die Aufbereitungsgewichte (deblurring weights) festgelegt sind, sind die Werte m(i,j) und A(i,j) konstant. Der Faktor 2 ^S spiegelt Präzisions-Zusatzbits wieder, welche später durch Stellenverschiebungen nach rechts in den Schritten (9.2) und (10) entfernt werden.

Durch die Einstellung auf A(0,0) wird eine Rundungsabweichung korrigiert, damit die Ausgangswerte darunter ohne eine Rundungskorrektur verwendet werden können. Wie hier angegeben, beruht A(0,0) auf der Addition von 1 zu L(0,0) beim Schritt (6). Die Interpolation "(25 − i − j)/64" ist heuristisch, ist aber annähernd optimal im Sinne eines mittleren quadratischen Fehlers. Nach einmal, die 20 verschachtelte Version.

9. Über das transformierte Bild wird iteriert, indem das Folgende an jedem (8 × 8)-Block von transformierten Luminanzwerten L(_,_) durchgeführt wird, was beim Schritt (5) hergeleitet worden ist.

9.1 Vorbereiten der Werte

$$E(i,j) = L(i,j) \cdot m(i,j) + A(i,j)$$
für L(i,j) > 0

$E(i,j) = L(i,j) \cdot m(i,j) - A(i,j)$
für $L(i,j) < 0$

$E(i,j) = 0$
für $L(i,j) = 0$
für jeweils (i,j), $i = 0, 1, 2, \ldots, 7$ und $j = 0, 1, 2, \ldots, 7$.

A(0,0) muß immer hinzuaddiert werden. Durch die Erfindung ist auch der Fall abgedeckt, bei welchem der Test $L(0,0) > 0$ nicht gemacht wird und die Schritte (6) und (8) (was freigestellt ist) vereinfacht werden. In der Praxis sollten kleine Multiplikationen, z. B. $-11 < L(i,j) < 11$ als Spezialfälle erkannt werden, um den Rechenaufwand einer Multiplikation zu sparen.

9.2 (Um gegebenenfalls die Kosten der Halbleitereinrichtung zu verringern, sind die Zahlen E(i,j) um eine beliebige Anzahl von Positionen S1 nach rechts zu verschieben. Diese Verschiebungen sind in einigen Ausführungen des Verfahrens "frei". Bei Ausführungen, wo das Verschieben nicht frei ist, wenn sie wahlweise weggelassen werden, wenn E(i,j) null ist, oder es können alle Verschiebungen wahlweise ausgeschlossen werden, indem $S1 = 0$ gesetzt wird).

9.3 In der zweidimensionalen Form sind noch einmal die Werte aufzubereiten:

$F(0,j) = E(4,j) + E(0,j)$
$F(4,j) = E(0,j) - E(4,j)$
$F(2,j) = Db \cdot E(6,j) + Nb \cdot E(2,j)$
$F(6,j) = Db \cdot E(2,j) - Nb \cdot E(6,j)$
$F(1,j) = Da \cdot E(7,j) + Na \cdot E(1,j)$
$F(7,j) = Da \cdot E(1,j) - Na \cdot E(7,j)$
$F(3,j) = Dc \cdot E(5,j) + Nc \cdot E(3,j)$
$F(5,j) = Dc \cdot E(3,j) - Nc \cdot E(5,j)$

$H(0,j) = F(0,j) + F(2,j)$
$H(1,j) = F(4,j) + F(6,j)$
$H(2,j) = F(4,j) - F(6,j)$
$H(3,j) = F(0,j) - F(2,j)$
$H(4,j) = F(7,j) - F(5,j)$
$H5(j) = F(1,j) + F(5,j)$
$H6(j) = F(1,j) - F(3,j)$
$H(5,j) = H6(j) + H5(j)$
$H(7,j) = F(1,j) + F(3,j)$
für $j = 0, 1, 2, \ldots, 7$

9.4 Aufbereiten der Werte:

$G(i,0) = H(i,4) + H(i,0)$
$G(i,4) = H(i,0) - H(i,4)$
$G(i,2) = Db \cdot H(i,6) + Nb \cdot H(i,2)$
$G(i,6) = Db \cdot H(i,2) - Nb \cdot H(i,6)$
$G(i,1) = Da \cdot H(i,7) + Na \cdot H(i,1)$
$G(i,7) = Da \cdot H(i,1) - NA \cdot H(i,7)$
$G(i,3) = Dc \cdot H(i,5) + Nc \cdot H(i,3)$
$G(i,5) = Dc \cdot H(i,3) - Nc \cdot H(i,5)$

$M(i,0) = G(i,0) + G(i,2)$
$M(i,1) = G(i,4) + G(i,6)$
$M(i,2) = G(i,4) - G(i,6)$
$M(i,3) = G(i,0) - G(i,2)$
$M(i,4) = G(i,7) - G(i,5)$
$M5(i) = G(i,7) + G(i,5)$
$M6(i) = G(i,1) - G(i,3)$
$M(i,5) = M6(i) - m5(i)$
$M(i,6) = M6(i) + M5(i)$
$M(i,7) = G(i,1) + G(i,3)$
für $i = 0, 1, 2, \ldots, 7$

9.5 Jedes M(i,j) multiplizieren mit

(falls $i = 0, 2, 3$ oder ist:)

Nrc', wenn $j = 5$ oder 6

Drc', wenn j = 4 oder 7
1 (keine Aktion) wenn j = 0, 1, 2 oder 4.

(falls i = 4 oder 7 ist)

Drr', Nrc', wenn j = 5 oder 6
Drr', DRC', wenn j = 4 oder 7
Drr', wenn j = 0, 1, 2 oder 3

(falls i = 5 oder 6:)

Nrr', Nrc', wenn j = 5 oder 6
Nrr', Drc', wenn j = 4 oder 7
Nrr', wenn j = 0, 1, 2 oder 3.

9.6 Aufbereiten der Werte:

$Z(i,0) = M(i,0) + M(i,7)$
$Z(i,1) = M(i,1) + M(i,6)$
$Z(i,2) = M(i,2) + M(i,5)$
$Z(i,3) = M(i,3) + M(i,4)$
$Z(i,4) = M(i,3) - M(i,4)$
$Z(i,5) = M(i,2) - M(i,5)$
$Z(i,6) = M(i,1) - M(i,6)$
$Z(i,7) = M(i,0) - M(i,7)$
für $i = 0, 1, 2, \ldots, 7$

9.7 Aufbereiten der Werte

$Y(0,j) = Z(0,j) + Z(7,j)$
$Y(1,j) = Z(1,j) + Z(6,j)$
$Y(2,j) = Z(2,j) + Z(5,j)$
$Y(3,j) = Z(3,j) + Z(4,j)$
$Y(4,j) = Z(3,j) - Z(4,j)$
$Y(5,j) = Z(2,j) - Z(5,j)$
$Y(6,j) = Z(1,j) - Z(6,j)$
$Y(7,j) = Z(0,j) - Z(7,j)$
für $j = 0, 1, 2, \ldots, 7$

10. Nach dem Schritt 9.7 in jedem Bild-Unterblock und für jede der 64 Stellen (i,j) ist der Wert aufzubereiten.

$$V(i,j) = Y(i,j) \cdot 2^{S1-S}$$

wobei S und S1 beliebige ganze Zahlen sind, welche bei den Schritten (7) und (9.2) definiert sind. Die Multiplikation ist in Wirklichkeit eine Verschiebung nach rechts.

11. In Abhängigkeit von Systembesonderheiten kann nunmehr eine Bereichüberprüfung notwendig sein. Wenn beispielsweise der zulässige Luminanzbereich $0 \leqq v(i,j) \leqq 255$ ist, dann sollten Werte von v(i,j), welche kleiner als 0 und größer als 255 sind, durch 0 bzw. 255 ersetzt werden. Die Werte v(i,j) sind die nunmehr rekonstruierten Bildluminanzwerte.

Diskussion von Sekundärprozessen

Üblicherweise wird der Primärprozeß durch zusätzliche Maßnahmen ergänzt, um die Verdichtung oder Bildqualität zu verbessern. Nach dem Schritt 10 kann die Bildgenauigkeit dadurch verbessert werden, daß alle Bildelementpaare V(8I+7,j), V(8I+8,j) und alle Bildelementpaare V(k, 8J+7), v(i, 8J+8) (d. h. benachbarte Bildelemente, welche auf gesonderte Bildblöcke aufgeteilt wurden) iteriert werden bzw. ihre Werte v1, v2 beispielsweise um

$$(v2 - v1)/\max(2, 11\sqrt{(M)})$$

inkrementiert bzw. dekrementiert werden, wobei M der Raten-Normierfaktor ist, welcher beim Schritt (4) verwendet worden ist und wobei der Ausdruck in dem Nenner wieder eine geeignete Annäherung an das Optimale ist.

Vor einem Durchführen des Schritts (6) kann die subjektive Schwierigkeit des lokalen Bildbereichs vorzugsweise in einen von drei Typen, nämlich eine einfache, eine doppelte oder eine vierfache Genauigkeit mit dem Vorsatzcode '0', '10', bzw. '11' klassifiziert werden. Die Rechnung beim Schritt (6) wird nunmehr ersetzt durch

$$L(i,j) = G(i,j) * K(i,j) * 2^{P-s(i,j)}$$

wobei p=0, 1 oder 2 für eine einfache, doppelte bzw. vierfache Genauigkeit steht. Dies wird später beim Schritt 9.2 kompensiert, bei welchem die zusätzliche Genauigkeit mit einer (größeren) Verschiebung nach rechts entfernt werden muß.

Leider ist kein sehr effektives, einfaches Klassifikationsschema gefunden worden. Gegenwärtig wird ein beschwerliches Schema verwendet, bei welchem die Schwierigkeits-Maßnahme P von vier Quellen abgeleitet wird:

a) P_links und P_oben, die Schwierigkeitsmaßnahmen von benachbarten Bildbereichen;
b) sum(i + j)G(i,j)'2)/sum(G(i,j)'2, der Transformations-Energie-Bitversatz (transform energy skew)
c) −G(0,0), die inverse mittlere Luminanz, und
d) max(Summe über festgelegte Breite (sum_over_fixed_with) (Histogramm (v(i,j)))), die Einheitlichkeit.

Beim Schritt (7) können die Transformationsdaten L(,), welche zu speichern oder zu übertragen sind, mit Hilfe einer Entropie-Codiermethode weiter reduziert werden. Die Anmelderin verwendet und empfiehlt eine Ausarbeitung des CCITT-Zick-Zack-Run-and-Template-Code mit mehreren vorgegebenen Huffman-Tabellen in Abhängigkeit von der Bitrate. Im folgenden Abschnitt wird ein derartiges Beispiel ausführlich behandelt.

Beispiel für ein verdichtetes (komprimiertes) Dateiformat

Ein verdichtetes Bild wird dargestellt durch

) Vorwort bzw. -spann (Bildbreite, -höhe, Ratennormierfaktor M, usw.)
2) Bildelementblock 0
Bildelementblock 1
Bildelementblock 2
...
Bildelementblock N — 1
3) Nachwort bzw. -spann (wenn überhaupt)

wenn jeder Bildelementblock dargestellt ist durch

1) einen Präzisionscode (welcher durch einen Zusatzschritt Z festgelegt worden ist)
2) einen Gleichstrom-Koeffizienten-Deltacode
3) einen Wechselstrom-Koeffizientencode (der null- oder mehrmals wiederholt wird)
4) einen Endblock-Code,

wobei jeder Wechselstrom-Koeffizientencode dargestellt ist durch

1) eine Neun-Null-Dehnung (die E-mal wiederholt worden ist, E 0)
2) einen Run-Template-Code, der (R, T) anzeigt
3) ein Vorzeichen eines Koeffizientenwerts (1 Bit)
4) einen Absolutwert des Koeffizienten, wobei das höchstwertige Bit gelöscht ist (T Bits)

wobei R+(*E die Anzahl an nullwertigen Koeffizienten ist, welche dieser Eins in einer "Zickzack"-Reihenfolge vorangehen (eine Folge, die auf der Summe i+j basiert) und wobei T die Bit-Position des höchstwertigen Bits (MSB) des Absolutwerts des Koeffizienten beispielsweise T=3 ist, wenn der Koeffizient 11 oder −11 ist:

Bitposition: 876543210
11 = 000001011 (Binär)
— höchstwertiges Bit

Eine Wahl oder ein Kodieren des Gleichstrom-Koeffizienten-Deltacodes wird im einzelnen nicht erläutert; jedoch wird ein Beispiel eines Huffman-Codes gegeben, welcher bei höheren Bitraten für den Wechselstrom-Run-And-Template-Code brauchbar ist.

Code	R	T
0xx	0	w
100x	0	4+w
111110	0	6
1111110{0}	0	7+n
1010	1	0
10110	1	1
10111	2	0
1100xx	1+w	max(0, 2−w)
11010{0}1xx	1+w	n+1+max(0, 2−w)
11011xx	5+w	0
111100{0}>1xx	1+w	n+1+max(0, 2−w)
11011xx	5+w	0
111100{0}>1xx	%+w	1+n
1111111		= reserviert
111101		= Neun-Null-Dehnung
1110		= Endblock-Code

wobei (0) n aufeinanderfolgende Nullen, n = 0, 1, 2, 3 . . . bezeichnet, xx 2 Bits bezeichnet, die als w = 0, 1, 2 oder 3 interpretiert werden, und x ein 1-Bit bezeichnet, das als w = 0 oder 1 interpretiert wird.

128-Punkt- und 256-Punkt-Transformationen

Die vorhergehende Methode kann bei einer größeren, verallgemeinerten Chen-Transformation, nämlich bei 8 mal 16 oder 16 mal 16, angewendet werden. Bei der Methode, die Chen-Transformation weiter zu verallgemeinern, sollte klar sein, daß die 1D-16-Punkt-GCT-Transformation (mit Reihen in "Schmetterlings-Ordnung" und ohne die notwendigen Normier-Nachmultiplikationen) gegeben ist durch:

GCT_16 (a, b, c, e, f, g, h, r, s, t) =

$$\begin{vmatrix} GCT\_8\ (a, b, c, r) & GCT\_*\ (a, b, c, r) \\ GQ\ (e, f, g, h, r, s, t) & -GQ\ (e, f, g, h, r, s, t) \end{vmatrix}$$

wobei

GCT8 (a, b, c, r) =

$$\begin{vmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ b & 1 & -1 & -b & -b & -1 & 1 & b \\ b & 1 & -1 & -b & -b & -1 & 1 & b \\ a & ar+r & ar-r & 1 & -1 & r-ar & -r-ar & -a \\ 1 & -cr-r & cr-r & c & -c & r-cr & cr+r & -1 \\ c & r-cr & -cr-r & -1 & 1 & cr+r & cr+r & -c \\ 1 & r-ar & ar+r & -a & a & -r-ar & ar-r & -1 \end{vmatrix}$$

und wobei GQ8 (e, f, g, h, r, s, t) =

Hierbei sind die "wahren Cosinus"-Parameter:

g = tan 15pi/32 ≈ 10,1532
a = tan 14pi/32 ≈ 5,0273
f = tan 13pi/32 ≈ 3,2966
b = tan 12pi/32 ≈ 2,4142
g = tan 11pi/32 ≈ 1,8709

c = tan 10pi/32 ≅ 1,4966
h = tan 9pi/32 ≅ 1,2185
r = cos 8pi/32 ≅ 0,7071
t = cos 12pi/32 ≅ 0,3827
s = cos 4pi/32 = t • b

Die verwendeten Parameter sind:

e = 10
a = 5
f = 3,25
b = 2,4
g = 1,875
c = 1,5
h = 1,25
r = 17/240,708333
r = 17/240,708333
t = 5/13 ≅ 0,384615
s = t • b = 12/13

Die Umkehr von GQ8 (e, f, g, h, r, s, t) ist die Transponierte von GQ8 (e, f, g, h, 1/2r, t' b, t') wobei b = s/t und t' = 1/t+t•b•b) ist.

Beispiel: Matrizen

Transponierte der Matrix TP

Die Cosinus-Transformation (a = 5,02734, b = 2,41421, c = 1,49661, r = 0,70711):

0,1768	0,1768	0,1768	0,1768	0,1768	0,1768	0,1768	0,1768
0,2452	0,2079	0,1389	0,0488	−0,0488	−0,1389	−0,2079	−0,2452
0,2310	0,0957	−0,0957	−0,2310	−0,2310	−0,0957	0,0957	0,2310
0,2070	−0,0488	−0,2452	−0,1389	0,1389	0,2452	0,0488	−0,2079
0,1768	−0,1768	−0,1768	0,1768	0,1768	−0,1768	0,1768	0,1768
0,1389	−0,2452	0,0488	0,2079	−0,2079	−0,0488	0,2452	−0,1389
0,0957	−0,2310	0,2310	−0,0957	−0,0957	−0,2310	−0,2310	0,0957
0,0488	−0,1389	0,2079	−0,2452	0,2452	0,2452	−0,2079	0,1389

Diesbezügliche Chen-Transformation (a = 5,0; b = 2,4; c = 1,5; r = 0,7)

0,1768	0,1768	0,1768	0,1768	0,1768	0,1768	0,1768	0,1768
0,2451	0,2059	0,1373	0,0490	−0,0490	−0,1373	−0,2059	−0,2451
0,2308	0,0962	−0,0962	−0,2308	−0,2308	−0,0962	0,0962	0,2308
0,2080	−0,0485	−0,2427	−0,1387	0,1387	0,2427	0,0485	−0,2080
0,1768	−0,1768	−0,1768	0,1768	0,1768	−0,1768	−0,1768	0,1768
0,1387	−0,2427	0,0485	0,2080	−0,2080	−0,0485	0,2427	−0,1387
0,0962	−0,2308	0,2308	−0,0962	−0,0962	0,2308	−0,2308	0,0962
0,0490	−0,1373	0,2059	−0,2451	0,2451	−0,2059	0,1373	−0,0490

Beschreibung einer Einrichtung

Nunmehr wird eine erfindungsgemäße Einrichtung im einzelnen beschrieben. Im folgenden wird der Ausdruck "Punkt" (point) verwendet, um ein Normier-(Scaler)Register oder einen Datenweg beliebiger Präzision, üblicherweise 8 bis 12 Bits zu bezeichnen. Ein Verfahren, um eine entsprechende Präzision festzusetzen, ist bekannt (siehe Jalali and Rao "Limited Wordlength and FDCT Processing Accuracy", "IEEE ASSP-81", Vol. III, Stn 1180-2).

Bei der Software-Methode werden die Transformationsstufen kombiniert, und die Wu-Paolini-Verbesserung wurde angewendet. Für die Halbleiter-Einrichtung ist es einfach bequemer, zwei 8-Punkt-Transformationseinheiten vorzusehen, und zwar jeweils eine für die vertikalen und die horizontalen Richtungen. Es ist eine 6-Punkt-Schiebeanordnung zwischen den vertikalen und horizontalen Transformationen vorzusehen und eine entsprechende Pufferung zwischen dem Transformationsabschnitt und dem Codierabschnitt.

Obwohl die Erfindung eine monochromatische Einrichtung und/oder gesonderte Einrichtungen für die Kompression und Dekompression vorsieht, hat eine bevorzugte Ausführungsform (**Fig.** 7) einen Kompressor-Verdichter (**Fig.** 1A) und einen Dekompressor (**Fig.** 1B), welche mit dreifarbigen Daten arbeiten. Daten werden dem Verdichter (**Fig.** 2A) in Vektoren von 8 Bildelementen zugeführt, welche ferner in einer lexikographischen Reihenfolge in Blöcken von 64 Bildelementen angeordnet sind. Die Blöcke werden einer Pipeline-Verarbeitung

unterzogen (Fig. 2B).

Ein Bildelement-Eingang an dem Verdichter weist "R"-(Rot-), "G"-(Grün-) und "B"-(Blau-)Normierungsfaktoren (Scalers) auf. Diese werden unmittelbar in einem Luminanz-Chrominanz-Raum transformiert (die Gründe für eine derartige Transformation sind bekannt).

Bei der Transformation können willkürliche fest oder programmierbare Koeffizienten (Fig. 3A) verwendet werden, oder es kann in einfachen Werten in einer dedizierten Anwendung (Fig. 3B) "fest verdrahtet" werden. Der Transformationsraum ist hier als XYZ bezeichnet; es kann auch irgendeine lineare Form der dreifarbigen Eingabe, vielleicht der CCITT-Standard: (Y, R—Y, B—Y), verwendet werden. Die drei Werte X, Y, Z sind dann jeweils gesonderten Schwarzweiß-(monochromen)Verdichtern zugeführt. Der Dekompressor benutzt dieselbe oder eine ähnliche Schaltungsanordnung wie in Fig. 3, außer daß nunmehr ein XYZ-Vektor in einen RGB-Vektor transformiert wird.

Die Werte Y, X und Z werden dann in drei Schieberegister (Fig. 5) eingegeben, um auf die Abgabe an die erste Transformationseinheit zu warten. Die Transformationseinheit wirkt auf 2,6 Bildelemente, so daß einige der Daten, wie dargestellt, stark verzögert werden. Die Bezeichnung "XYZ" ist ein bißchen unglücklich; bei optimierten Codierverfahren muß die Luminanz ("Y") zuerst verarbeitet werden.

Während des Dekomprimierens ist das Problem bei dem XYZ-Versatz umgekehrt. In der bevorzugten Ausführungsform werden dadurch, daß während des Dekomprimierens die Benutzung von Y- und Z-Schieberegistern umgekehrt wird, 5 Registerstellen gespart.

In Fig. 1A weisen die Hauptabteilungen des Verdichters einen Eingabeabschnitt (1, 2) auf, welcher die Eingangswerte in den XYZ-Raum transformiert und ihn für einen anschließenden Transfer an die Transformationseinheit 3 puffert. Für jeweils acht Bildelement-Zeiten muß die Transformationseinheit dreimal zyklisch durchlaufen werden (einmal für jeden X-, Y- und Z-Datenwert). Der Ausgangswert von der Transformationseinheit 3 wird an das Schiebearray 4 angelegt, wo es zurückgehalten wird, bis der 8 × 8-Bildelementenblock vollständig gelesen worden ist. Die zweite Transformationseinheit (5, 6) verarbeitet den vorher gelesenen Bildelementblock, der sie wieder dreimal zyklisch durchläuft und liefert Daten an den Codierer-Eingabepuffer (7, 8). Der Codierer (9, 10, 11) wird für die drei Farbkoordinaten gemeinsam benutzt; jedoch wird der gesamte Luminanzblock ohne Unterbrechung codiert. Anschließend wird jeder der Chrominanzblöcke verarbeitet. Wenn die Verarbeitung dieser drei Blöcke innerhalb von 64 Bildelement-Zeiteinheiten nicht beendet werden kann, hält eine Synchronisier- und Steuerlogik den Bildelementtakt an der externen Eingabeschaltung an. Die Speicherbereiche (in dem Eingangsschieberegister 2, dem Schiebearray 4 und den Codierer-Eingabepuffern (7, 8) muß für die drei Farben verdreifacht werden: die Recheneinheiten (3, 5, 6, 9, 10, 11) werden (im Zeitmultiplex) von den Y-, X- und Z-Daten gemeinsam genutzt.

Der Codierer (9, 10, 11), der Codierer Eingabepuffer (7, 8), die Code-Programmiereinheiten (12, 13, 14) und eine (nicht dargestellte) Synchronisier- und Steuerlogik können gemäß dem bestehenden Stand der Technik folgen. Genauso ist das Verfahren für ein Zeit-Multiplexen von drei Farben durch eine einzige Schaltung bekannt. Ebenso sind der 3-Stellen-Transformationsabschnitt (1; Fig. 3) und die Schieberegister (2; Fig. 5) bekannt.

Die Normiereinheit (Scaler) 6 benutzt einen programmierten RAM oder ROM und ein System von (impliziten) Schiebeeinheiten, Multiplexern und Addierern. Dies ergibt eine unkomplizierte Realisierung. Aufgrund der Definition der verallgemeinerten Chen-Transformation und der entsprechenden Parameter ist auch die Ausführung des 8-Stellen-Transformators (Fig. 8) unkompliziert.

Das Schiebe-Array (Fig. 6A) wird nachstehend im einzelnen erläutert. Vertikale (transformierte) Vektoren aus dem laufenden Eingabe-Bildelementeblock werden zusammengesetzt, während horizontale Vektoren aus dem vorherigen Bildelementblock an den horizontalen Transformator oder Wandler übergeben werden. Ohne eine spezielle Ausführung würde dies 128 Register erfordern (und zwar jeweils 64 für den augenblicklichen und den vorherigen Block), da die Punkte in einer Reihenfolge verwendet werden, welche sich von der empfangenen Reihenfolge unterscheidet. Dies ist jedoch nicht notwendig, da die Daten während der geradzahligen Bildelement-Blöcke von links nach rechts und während der ungradzahligen Bildelement-Blöcke von oben nach unten geschoben werden. Das beschriebene Shift-Array ist zwei-gerichtet. Ein vier-gerichtetes Schiebearray wird in einigen Ausführungsformen bevorzugt.

In Fig. 6B ist im einzelnen das Schiebe-Array der Fig. 6A dargestellt. In Fig. 6B werden Vektoren aus dem Schiebe-Array an dessen Unterseite einzeln entnommen und an den DCT8-Abschnitt der Fig. 1A abgegeben. Inzwischen werden vertikale Vektoren von dem anderen DCT8-Abschnitt von oben in das Schiebe-Array eingegeben. Allmählich werden dadurch die alten Vektoren aus dem Schiebe-Array entfernt und das Schiebe-Array wird vollständig mit vertikalen Vektoren von dem nächsten Bildelementblock gefüllt.

Für den nächsten Bildelementblock unterscheidet sich die Datenflußrichtung um 90° bezüglich der Datenflußrichtung in einem vorhergehenden Bildelementblock. Auf diese Weise werden die horizontalen Vektoren auf der rechten Seite der Schiebe-Anordnung entnommen und an den DCT8-Abschnitt abgegeben, während die neuen vertikalen Vektoren der linken Seite hereinkommen. Bei einem Block N+2 wird durch eine weitere 90°-Drehung zu der Ausgangsform zurückgekehrt usw.

Der Dekompressor (Fig. 1B) hat einen Aufbau, welcher demjenigen des Verdichters (in Fig. 1A) ziemlich entspricht, außer daß die Datenflußrichtung umgekehrt wird. In einer bevorzugten Ausführungsform arbeitet eine einzige Einrichtung in zwei Betriebsarten, nämlich entweder als Verdichter bzw. Kompressor oder als ein Dekompressor.

Mögliche VLSI-Auslegungen (Fig. 4A, 4B) führen zu verschiedenen Datenflüssen für eine Verdichtung (Fig. 4A, 4B) und für eine Dekompression (Fig. 4A, 4B). Der Betrieb der Transformations- und Schiebe-Array-Einheiten hat dieselbe Richtung für die Kompression und für die Dekompression in der einen Auslegung (Fig. 4B), aber nicht in der anderen Auslegung (Fig. 4A). Dies ist deutlicher zu sehen, wenn der kombinierte

Kompressor-/Dekompressor-Datenfluß (**Fig.** 7) in Betracht gezogen wird. Wenn die zwei Transformationseinheiten den RGB- bzw. verdichteten Daten zugeordnet werden, ergeben sich bei der Auslegung Schwierigkeiten, wenn nicht ein vier-gerichtetes Schiebe-Array verwendet wird. Folglich sind die zwei Transformationseinheiten mit den Eingangs- bzw. Ausgangsabschnitten des Schiebe-Array verbunden.

In einer Ausführungsform benutzt die Transformationseinheit, welche in dem Kompressor verwendet ist (**Fig.** 8A), 38 Addiereinheiten. Ein Verschieben nach rechts um eine ("R1")-, zwei ("R2")- oder vier ("R4")-Positionen oder nach links um eine ("L1")-Position ist leicht zu machen. Die beschriebene Schaltung verwendet die Parameter (a, b, c, r) = (5, 2, 4, 1, 5, 17/24). Eine Ausführung mit b = 2,5 würde in einer anderen Ausführungsform nur 36 Addierglieder erfordern.

Eine entsprechende Schaltung ist für die inverse Transformationseinheit in dem Dekompressor erforderlich. Bei einer sorgfältigen Benutzung einer 'Ausgabe-Freigabe'-Signalisierung können die meisten der Addierglieder in dem Vorwärts-Transformator wieder verwendet werden. Eine solche Realisierung ist für einen Fachmann einfach.

Die Normiereinheit (Scaler) benutzt einen programmierten RAM oder ROM und ein System von impliziten Schiebeeinheiten, Multiplexern und Addierern. Auch dies ist einfach realisierbar. Die sogenannte Denormiereinheit (descaler) kann auf verschiedene Weise realisiert werden, vorzugsweise als eine kleine festverdrahtete Multipliziereinheit mit RAM, Akkumulator, Synchronisier- und Steuerlogik usw. In einer dedizierten preiswerten Anwendung kann die sogenannte Denormiereinheit (descaler) dadurch vereinfacht werden, daß Aufbereitungs-(deblurring)Gewichte bzw. Wertigkeiten über einen breiten Bereich nahezu optimal sind; folglich kann eine einfache Normierung, wie in der Normiereinheit verwendet werden. Die Denormiereinheit (descaler) kann entweder zwischen dem Codierer und seinem Ausgangspuffer oder zwischen dem Ausgangspuffer und einem Transformator angeordnet sein, wie in **Fig.** 1 und 7 dargestellt ist. Der Codierer-Eingangspuffer kann auf verschiedene Weise realisiert werden, so beispielsweise als eine Register-Reduzieranordnung mit einem zyklischen Sharing, was dem Schiebe-Array entspricht. Bei einer unkomplizierten Ausführung ist ein 384 mal 10-Bit-RAM mit einem 64 mal 7-Bit-ROM verwendet, um die RAM-Adressen zu schaffen.

Ein Beispiel eines Betriebszyklus wird nunmehr in Verbindung **Fig.** 1A und 1B beschrieben. In **Fig.** 1A werden Daten in den Verdichter als eine Dreifarben-Information, d. h. als rot, grün und blau eingegeben. Sie werden unmittelbar in einem Ausweichraum transformiert, welcher als XYZ bezeichnet ist. Die drei Elemente X, Y und Z werden jeweils in ihr eigenes Schieberegister eingegeben.

Von dem Schieberegister (Schritt 2) gehen sie in eine 8-Stellen-DCT-Einheit. Dies könnte entweder eine 8-Stellen-DCT-Einheit sein, welche unter den drei Farben X, Y und Z multiplex betrieben wird, oder sie können jeweils ihre eigene, individuelle DCT-8-Einheit haben.

Information wird dann in das 64-Stellen-Schiebe-Array 4 eingegeben. Dies ist ein individuelles Schiebe-Array für jede Farbe. Von dem Schiebe-Array (Block 4) geht es in eine andere DCT-Einheit (Block 5), welche dem Block 3 entspricht. Die Information ist dann so normiert, daß sie eine zusätzliche Schicht einer zusätzlichen Verschiebung ist. Die Information ist nur sowohl horizontal als auch vertikal transformiert. Das Schiebe-Array dreht die Daten begrifflich tatsächlich um 90°, so daß sie nunmehr in die andere Richtung transformiert werden können. Nachdem die Daten normiert sind, gehen sie in einen weiteren Puffer, welcher mit Blöcken 7 und 8 (Z1 und Z2) bezeichnet ist, um die Daten zu halten, so daß sie schließlich decodiert und von dem Chip aus ausgegeben werden können (wobei Z1, Z2 einem Zickzack gleichkommt).

Begrifflich entspricht dies dem Shift-Array (Block 4), außer daß nunmehr die Daten nicht um 90° gedreht werden. Statt dessen werden sie in eine Zickzack-Reihenfolge umgewandelt, welche üblicherweise für diese Dinge verwendet wird, und wird dann der CCITT-Standard benutzt. Die Information wird dann in dem "Run And Template"-Steuerblock dargestellt, welcher Nullen feststellt und Durchläufe für die Nullen erzeugt, welcher Nicht-Nullen feststellt und eine Schätzung des Logarithmus des Wertes erzeugt, welcher als Schablone (template) bezeichnet wird. Die Kombination "Run and Template" wird in einem RAM oder einem ROM abgelegt, was als der RT-Code bezeichnet wird, und wird dann von dem Chip aus ausgegeben.

Die Mantisse, welche den signifikanten Bits der Transferkoeffizienten entspricht, wird ebenfalls von dem Chip aus abgegeben. Da die Mantisse und der "Run And Template"-Code beliebig lang sind, ein Bit, zwei Bits, was auch immer, und der Ausgang von dem Chip immer 16 Bits oder 8 Bit, 32 Bit, was auch immer, sind, erleichtert dies ein (Ausricht-)Block 11.

Die anderen in **Fig.** 1A dargestellten Blöcke, (wahlweise) Programmierblöcke 12, 13 und 14 sind vorgesehen, um eine beliebige RGB- in eine XYZ-Transformation, beliebige Raten-Normierungen und psychoadaptive Wertungen sowie einen beliebigen, modifizierten Huffman-Code für den "Run And Template"-Prozeß durchzuführen. **Fig.** 1B ist der **Fig.** 1A sehr ähnlich. Der "Run And Template"-Code muß nunmehr in eine "Run And Template"-Kombination decodiert werden, und die notwendige Anzahl Nullen ist wegzulassen. In **Fig.** 1A ist die Normiereinheit eine sehr einfache Anordnung von Addierern und Schieberegistern. In **Fig.** 1B ist die Denormiereinheit (descaler) als eine sehr kleine Multipliziereinheit in Hardware ausgeführt.

In **Fig.** 9 ist ein Diagramm für eine zweidimensionale, verallgemeinerte Chen-Transformation hergestellt. Bildelemente kommen von oben herein und sind üblicherweise 8 Bits groß. Die Bildelemente durchlaufen eine große Anordnung von Addierern in dem horizontalen Umformblock 10 mit einer Datenbreite von üblicherweise 128 Bits; die Ausgangsdaten von dem horizontalen Umformblock 10 durchlaufen ein Transpositions-RAM 12, um die Information von horizontal in vertikal zu drehen. Die Daten laufen dann wieder in den vertikalen Umformblock 16, welcher wiederum nur Addierer (üblicherweise 128 Bits groß) aufweist. Die Ausgangskoeffizienten werden schließlich in ihrer Breite auf ungefähr 16 Bit reduziert und durchlaufen dann eine einzige Multipliziereinheit 20, welche gemäß der Erfindung JPEG-kompatibel ist.

In **Fig.** 10 ist ein Blockdiagramm für eine VLST-Ausführung gemäß der Erfindung dargestellt. In **Fig.** 10 kommen die Daten an einem Block 40 an und werden in dem Eingangs-Signalspeicher 42 gehalten und gelangen

über einen Multiplexer 44 in die erste Hälfte eines GCT-Transformationsblocks 50 (welcher ein Addierernetz ist). Die zweite Hälfte des Addierernetzes 60 ist rechts von den Mittelstufen-Haltegliedern (midstage latches) 54. Deren Ausgangsdaten laufen über ein MUX 62 in den Transpositions-RAM 66, in welchem die horizontale in die vertikale Transformation durchgeführt wird. Der Ausgang des Transpositions-RAM 66 ist um die erste Stufe des GCT-Blocks 50 herum rückgekoppelt, um die erste Hälfte der vertikalen Transformation in einer Zeit-Sharing- oder einer Zeitschachtelungs-Anordnung durchzuführen. Die Ausgangsdaten des GCT-Blocks werden an den Eingang der zweiten Stufe der vertikalen Transformationseinheit 60 angelegt. Schließlich werden die Ausgangsdaten des GCT-Blockes durch den Multiplexer 70 geleitet und laufen über eine Multipliziereinheit 74 und eine Rundungseinheit 76 in eine Arrangiereinheit 80 zur Bildung der Zickzack-Reihenfolge; an deren Ausgang wird ein 12-Bit-Koeffizient 84 abgegeben.

Anhand von Fig. 10 wird nunmehr der inverse Transformationsprozeß gemäß der Erfindung kurz beschrieben. In Fig. 10 werden die 12-Bit-Koeffizienten über einen Block 87 an den Y-Eingang der die Zickzack-Reihenfolge arrangierenden Einheit 80 eingegeben. Die Ausgangssignale der Arrangiereinheit 80 durchlaufen die Multipliziereinheit 74 und die Rundungseinheit 76, in welchem ein inverser Quantisierungsprozeß durchgeführt wird, welcher dem Prozeß entspricht, welcher bei dem Vorwärtsprozeß durchgeführt worden ist. Die Ausgangssignale der Multipliziereinheit 74 werden in den Signalspeicher 42 eingegeben, welcher die erste Stufe des inversen Transformationsprozesses ist. Von dem Signalspeicher 42 folgt der inverse Transformationsprozeß demselben Zweistufen-Zeitmultiplexweg, welchem bei dem Vorwärtsprozeß gefolgt worden ist. Die Ausgangssignale erscheinen an den Ausgabe-Haltegliedern 70, deren Ausgang Bildelemente sind, welche durch die Rundiereinheit 76 gerundet werden, deren Ausgang dem Block 40 zugeführt wird.

Vorstehend ist nur eine bevorzugte Ausführungsform der Erfindung beschrieben. Die Erfindung ist auch mit vorhandenen Standards, wie JPEG kompatibel. Die bevorzugte Ausführungsform wurde gewählt und beschrieben, um so das Prinzip der Erfindung und deren praktische Anwendungsmöglichkeiten zu erläutern, um dadurch anderen Fachleuten eine bestmögliche Benutzung der Erfindung aufzuzeigen.

Patentansprüche

1. Einrichtung zum Verdichten von Bildern, **gekennzeichnet durch**
eine horizontale Transformations-Einrichtung, um eingegebene Bildelemente mit einer bestimmten Bitbreite aufzunehmen und um die eingegebenen Bildelemente nur mit einer Addierer-Anordnung horizontal zu transformieren;
einen Transpositionsspeicher, um die horizontalen, transformierten Bildelemente in vertikale zu drehen und eine einzige Multipliziereinheit, um die transformierten, vertikalen Bildelemente aufzunehmen und um eine einzige Multiplikation an den transformierten vertikalen Bildelementen durchzuführen, um so verdichtete Bildelementdaten zu schaffen, welche die eingegebenen Bildelemente darstellen.

2. Verfahren zum Verdichten von Bildern, dadurch gekennzeichnet, daß eingegebene Bildelemente mit einer bestimmten Bitbreite aufgenommen und nur mit einer Addiereranordnung horizontal transformiert werden;
die horizontal transformierten Bildelemente in vertikale gedreht werden;
die vertikalen Bildelemente aufgenommen und nur mit einer zweiten Addiereranordnung vertikal transformiert werden, und
die transformierten, vertikalen Bildelemente aufgenommen und an den transformierten, vertikalen Bildelementen eine einzige Multiplikations-Funktion durchgeführt wird, um verdichtete Bildelementdaten zu schaffen, welche die eingegebenen Bildelemente darstellen.

3. Bildverdichtungssystem, gekennzeichnet durch
eine Einrichtung zum Aufnehmen von eingegebenen Bildelementdaten, welche ein Bild darstellen, und
eine Einrichtung zum Durchführen einer verallgemeinerten Chen-Transformation (GCT), um die Bilddaten zu verdichten,
wobei die GCT-Einrichtung eine GCT-Addiereinrichtung, um die Bilddaten nur mit Addierern horizontal zu transformieren, und einen Transpositionsspeicher aufweist, um die horizontal transformierten Bildelemente in vertikale zu drehen,
wobei die GCT-Addiereinrichtung eine Einrichtung, um die vertikalen Bildelemente nur mit Addierern vertikal zu transformieren, und eine Multipliziereinheit aufweist, um eine Multiplikationsfunktion an den transformierten, vertikalen Bildelementen durchzuführen, um verdichtete Bildelementdaten zu schaffen, welche die eingegebenen Bildelemente darstellen.

4. Bildverdichtungssystem nach Anspruch 3, dadurch gekennzeichnet, daß die GCT-Addiereinrichtung eine erste GCT-Addierer-Netzwert-Stufe, um die erste Hälfe der horizontalen und vertikalen Transformationen umzuarbeiten, und eine zweite GCT-Addierer-Netzwerk-Stufe aufweist, um die zweite Hälfte der horizontalen und vertikalen Transformationen umzuarbeiten.

5. Bildverdichtungssystem nach Anspruch 4, dadurch gekennzeichnet, daß die ersten und zweiten Einrichtungen die Bildelemente in einer Zeit-Sharing-Anordnung horizontal und vertikal transformieren.

6. Bildverdichtungssystem nach Anspruch 5, dadurch gekennzeichnet, daß die Multipliziereinheit eine Zickzack-Reihenfolge arrangierende Einrichtung aufweist.

7. Bildverdichtungssystem nach Anspruch 6, dadurch gekennzeichnet, daß die Multipliziereinheit eine Rundungseinrichtung aufweist.

8. Bildverdichtungssystem nach Anspruch 7, dadurch gekennzeichnet, daß die Multipliziereinheit eine Multipliziertabellen-Einrichtung aufweist.

Hierzu 12 Seite(n) Zeichnungen

Nummer: DE 41 33 460 A1
Int. Cl.[5]: G 06 F 15/66
Offenlegungstag: 15. April 1993

FIG. 1A

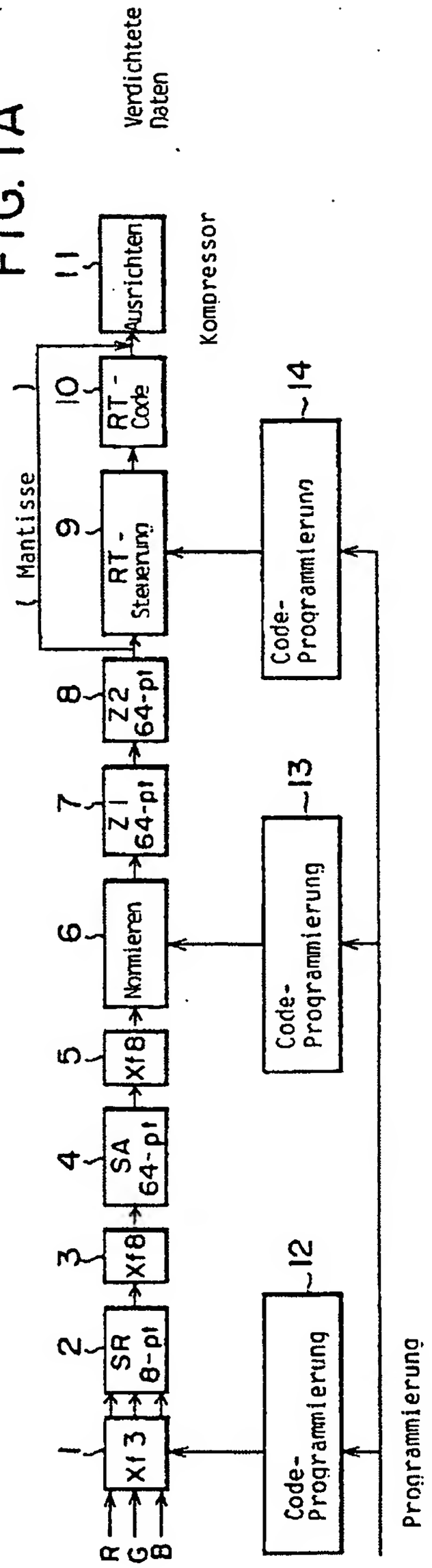
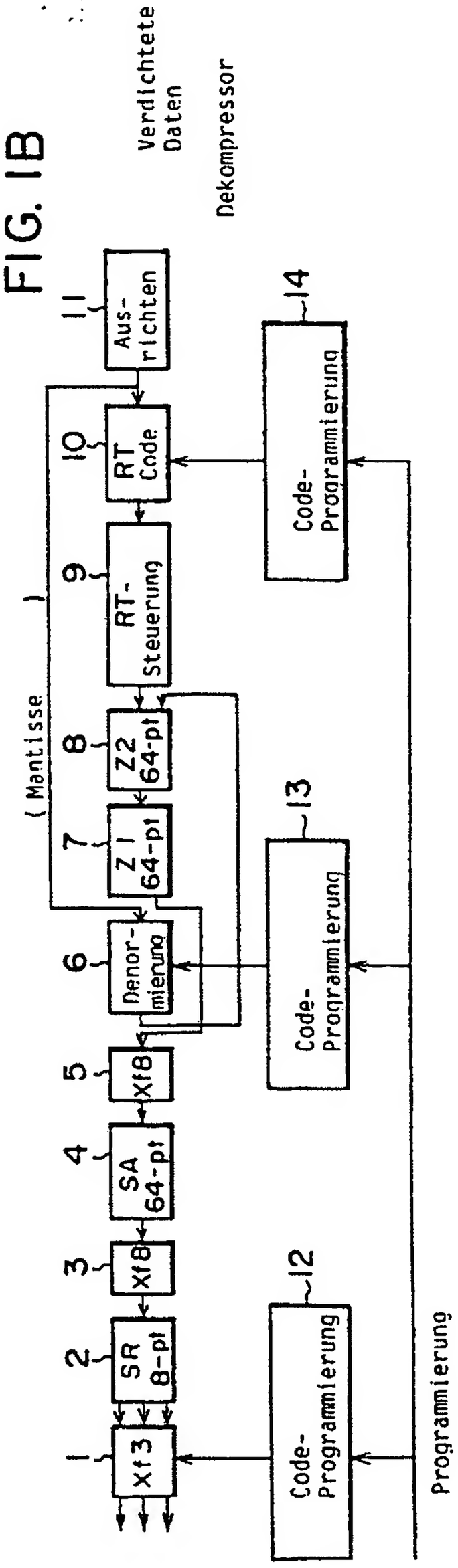


FIG. 1B



Nummer: DE 41 33 460 A1
Int. Cl.[5]: G 06 F 15/66
Offenlegungstag: 15. April 1993

FIG. 2A

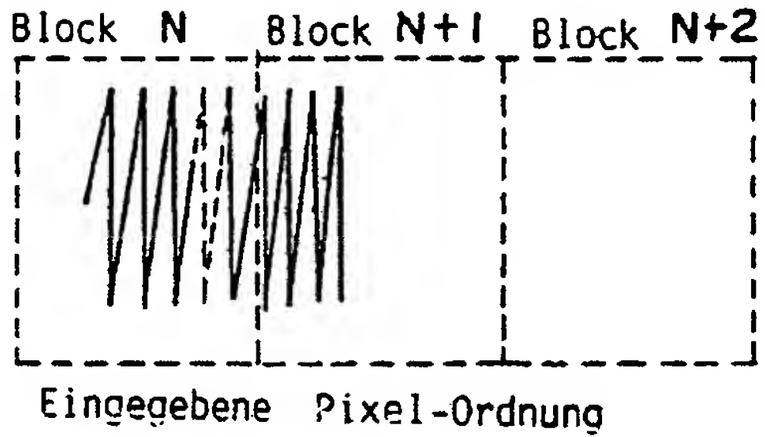


FIG. 2B

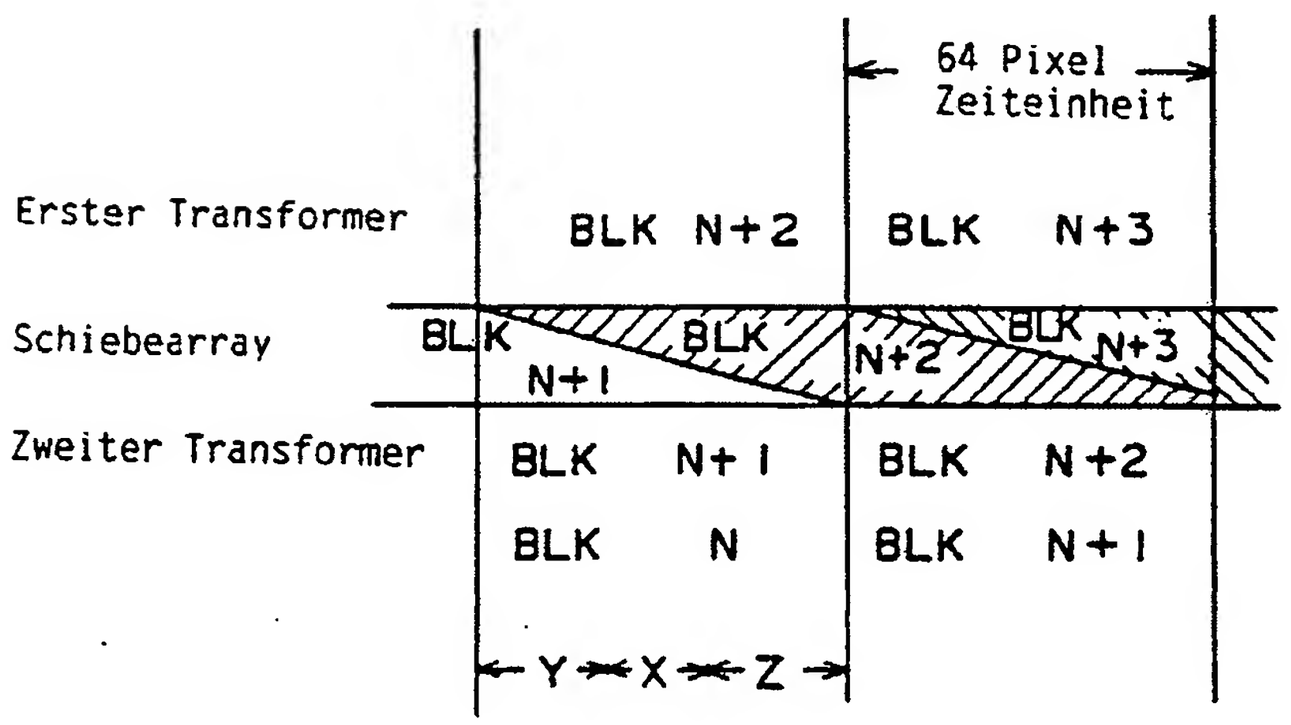
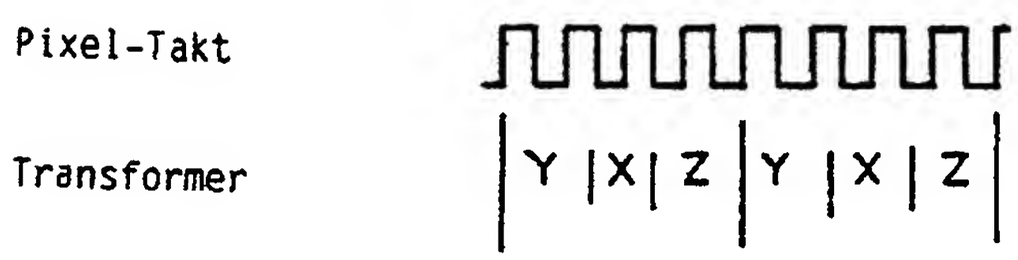


FIG. 2C



Nummer: DE 41 33 460 A1
Int. Cl.[5]: G 06 F 15/66
Offenlegungstag: 15. April 1993

FIG. 3A

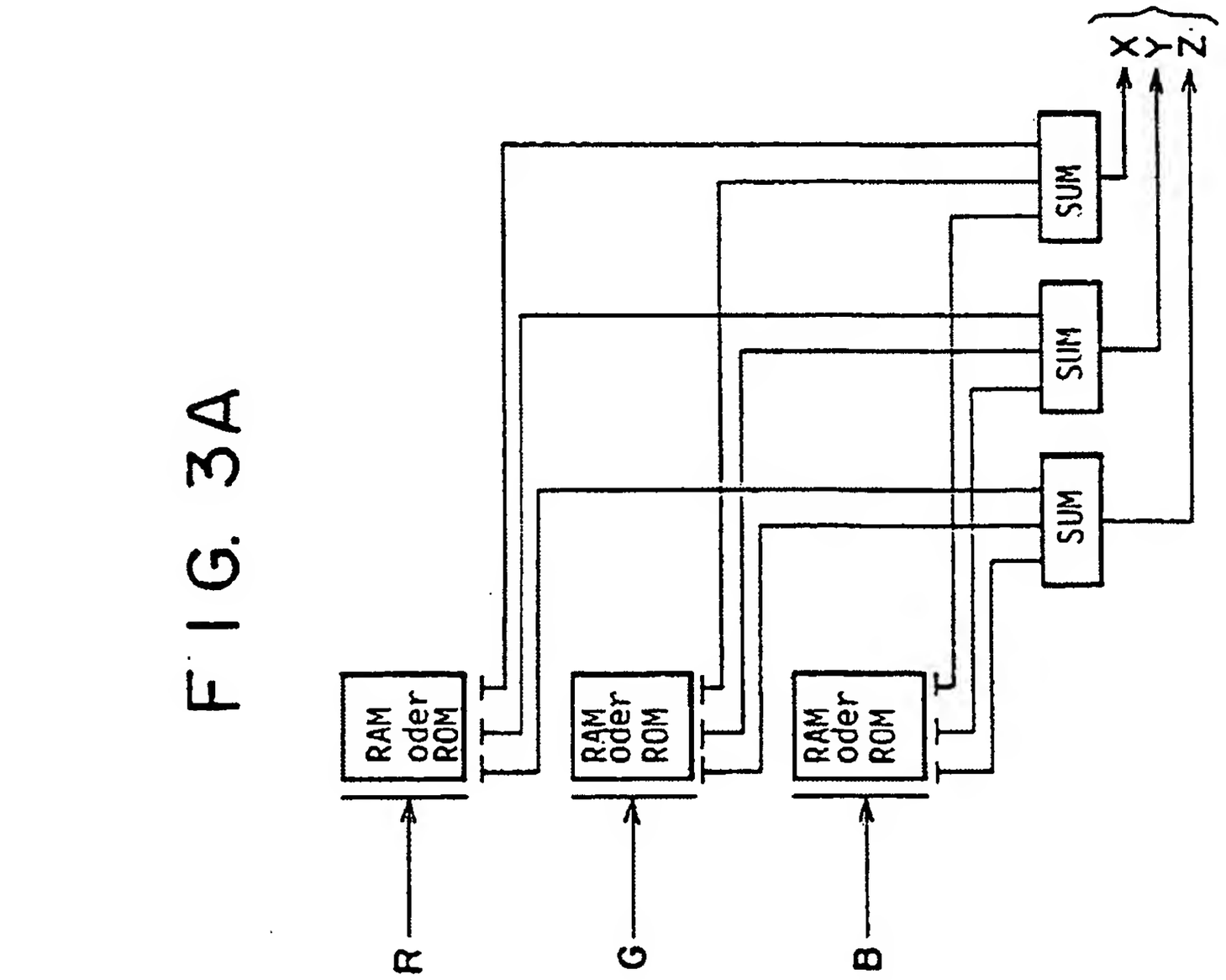
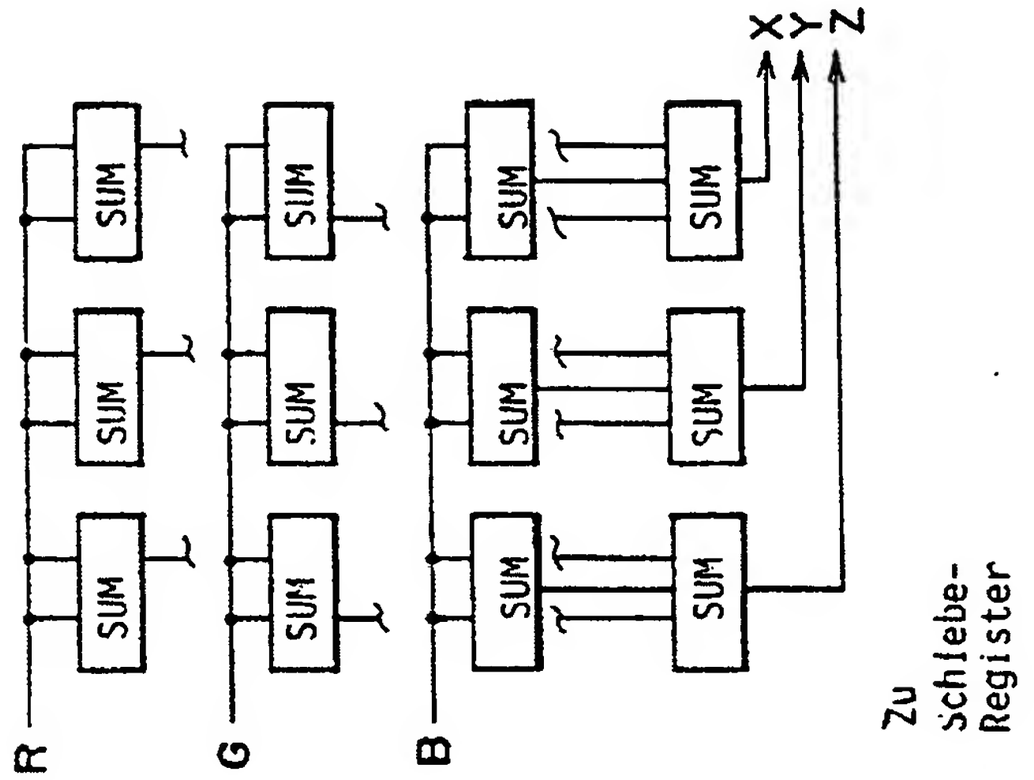


FIG. 3B



Nummer: DE 41 33 460 A1
Int. Cl.[5]: G 06 F 15/66
Offenlegungstag: 15. April 1993

FIG. 4A

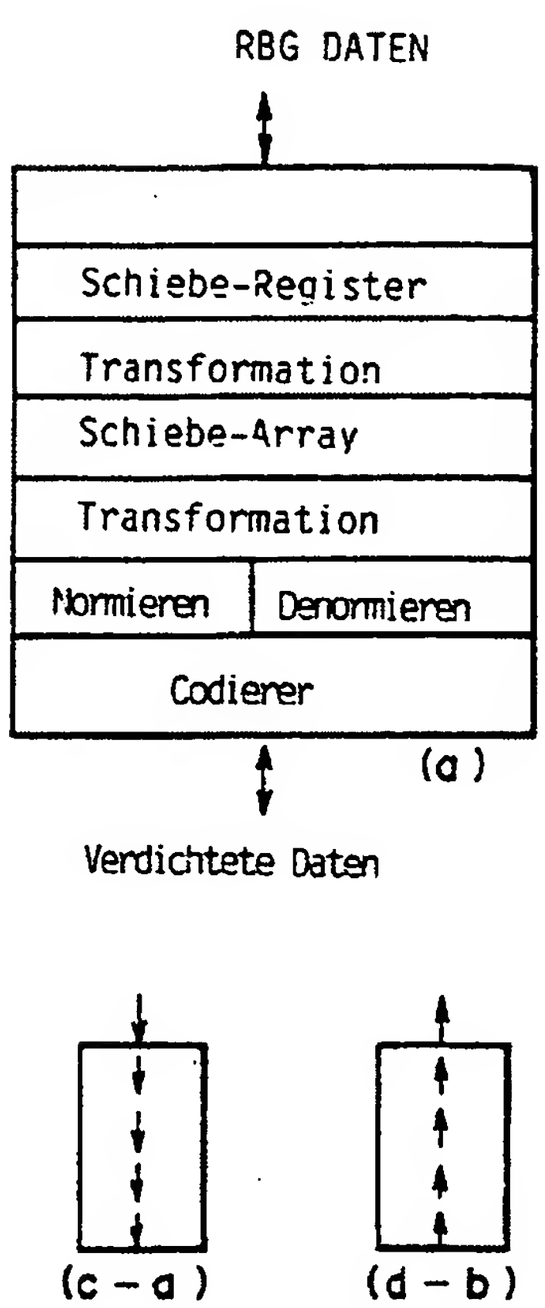
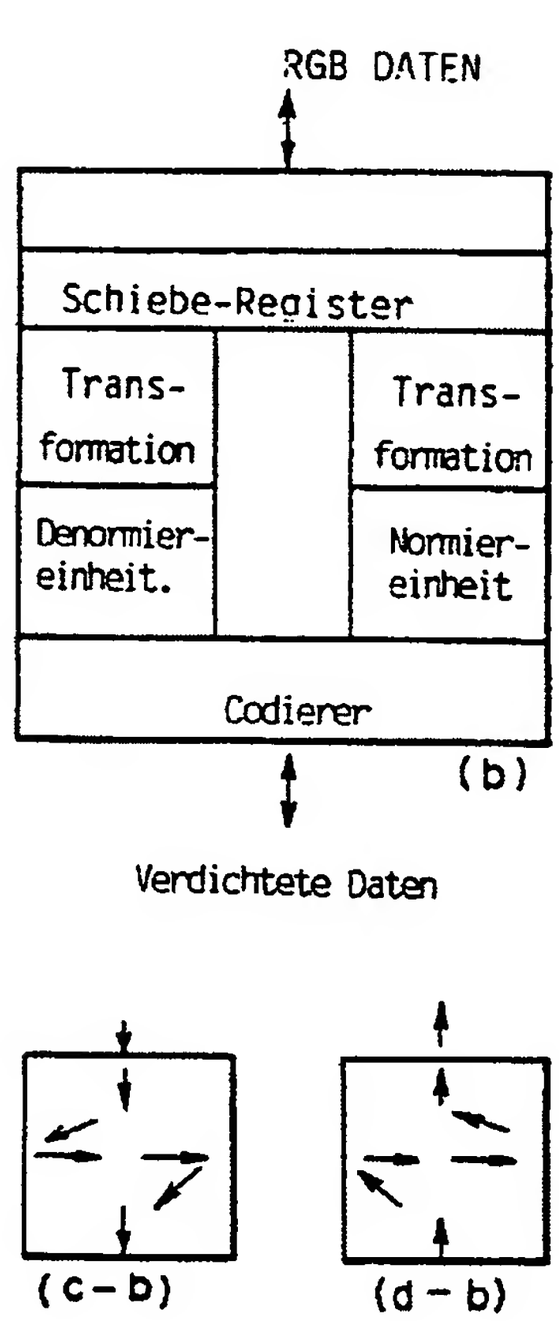
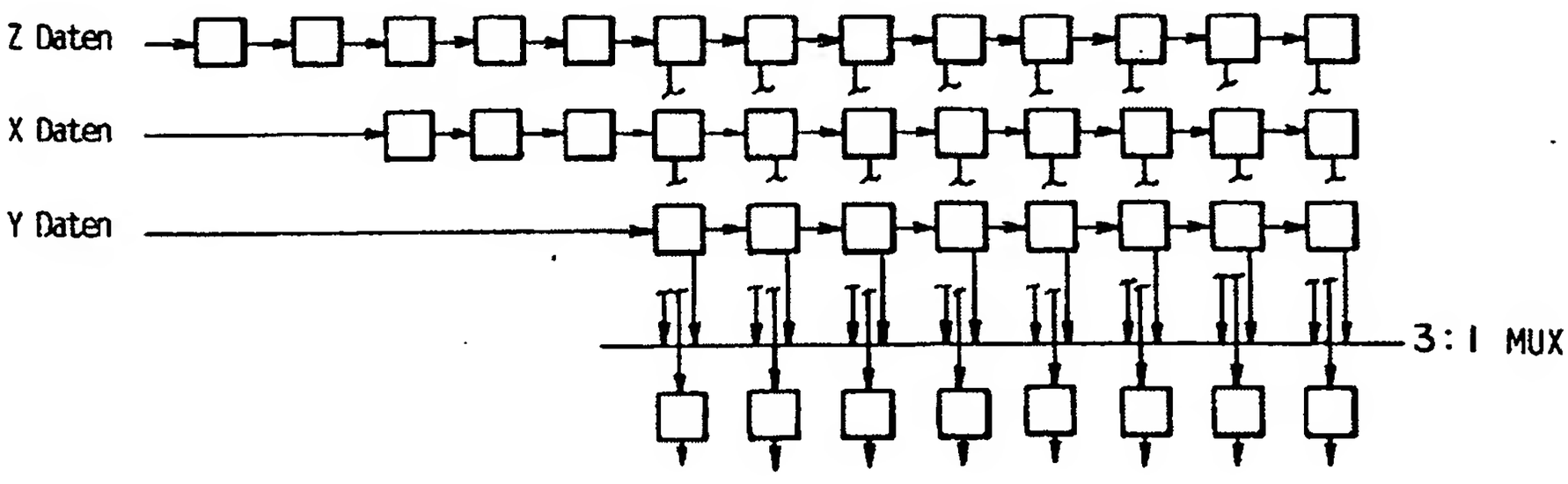


FIG. 4B



Nummer: DE 41 33 460 A1
Int. Cl.[5]: G 06 F 15/66
Offenlegungstag: 15. April 1993

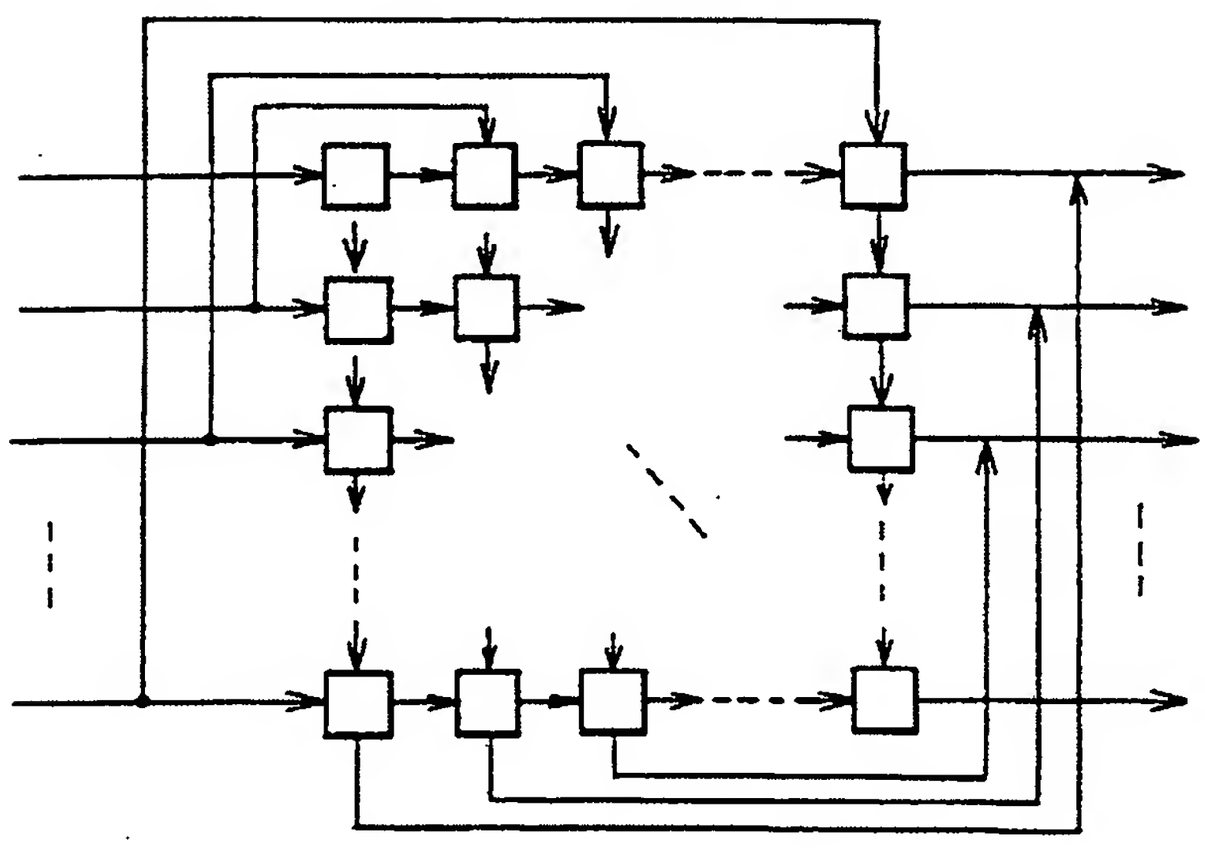
FIG. 5



Zur Transformation

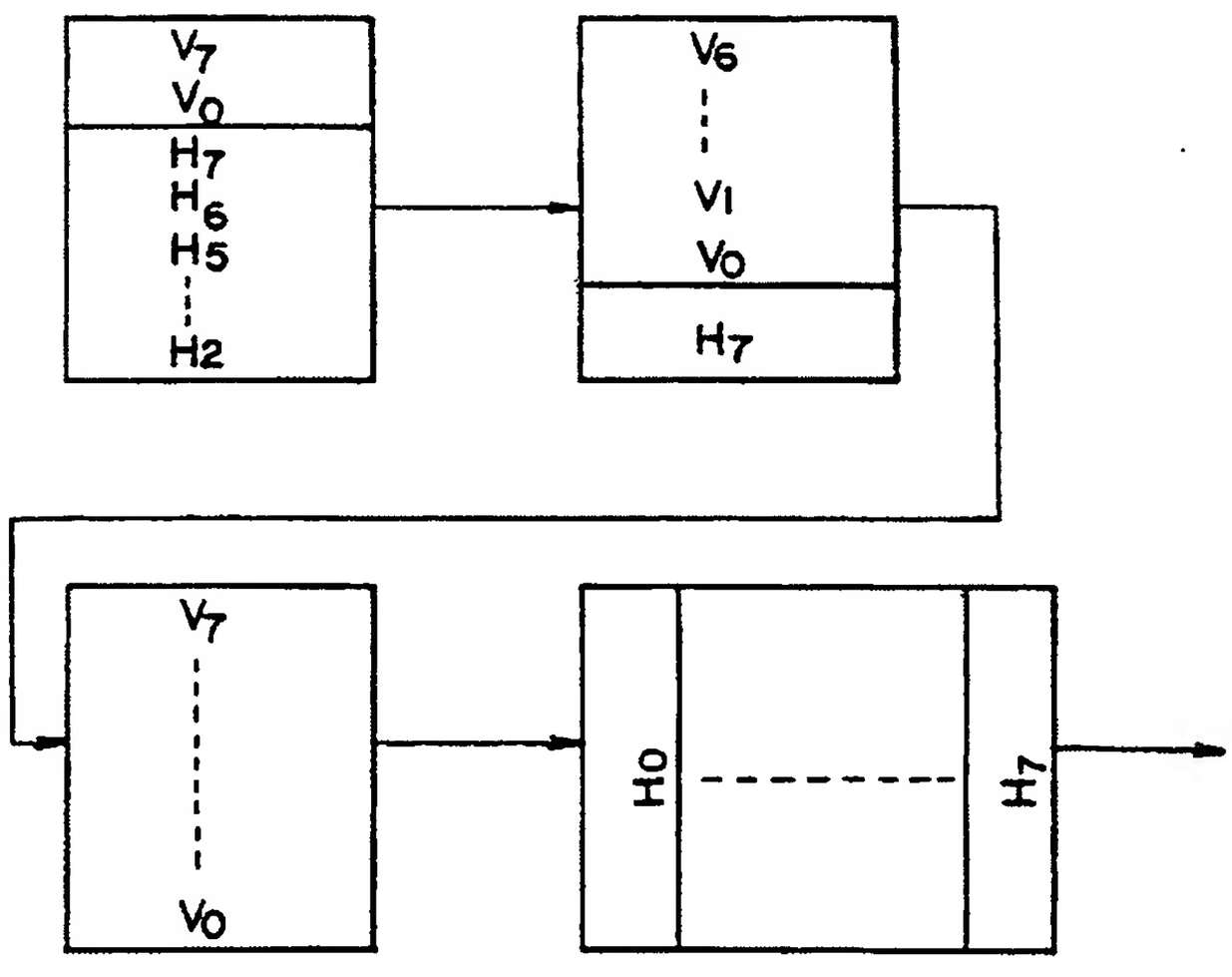
Nummer: DE 41 33 460 A1
Int. Cl.[5]: G 06 F 15/66
Offenlegungstag: 15. April 1993

FIG. 6A



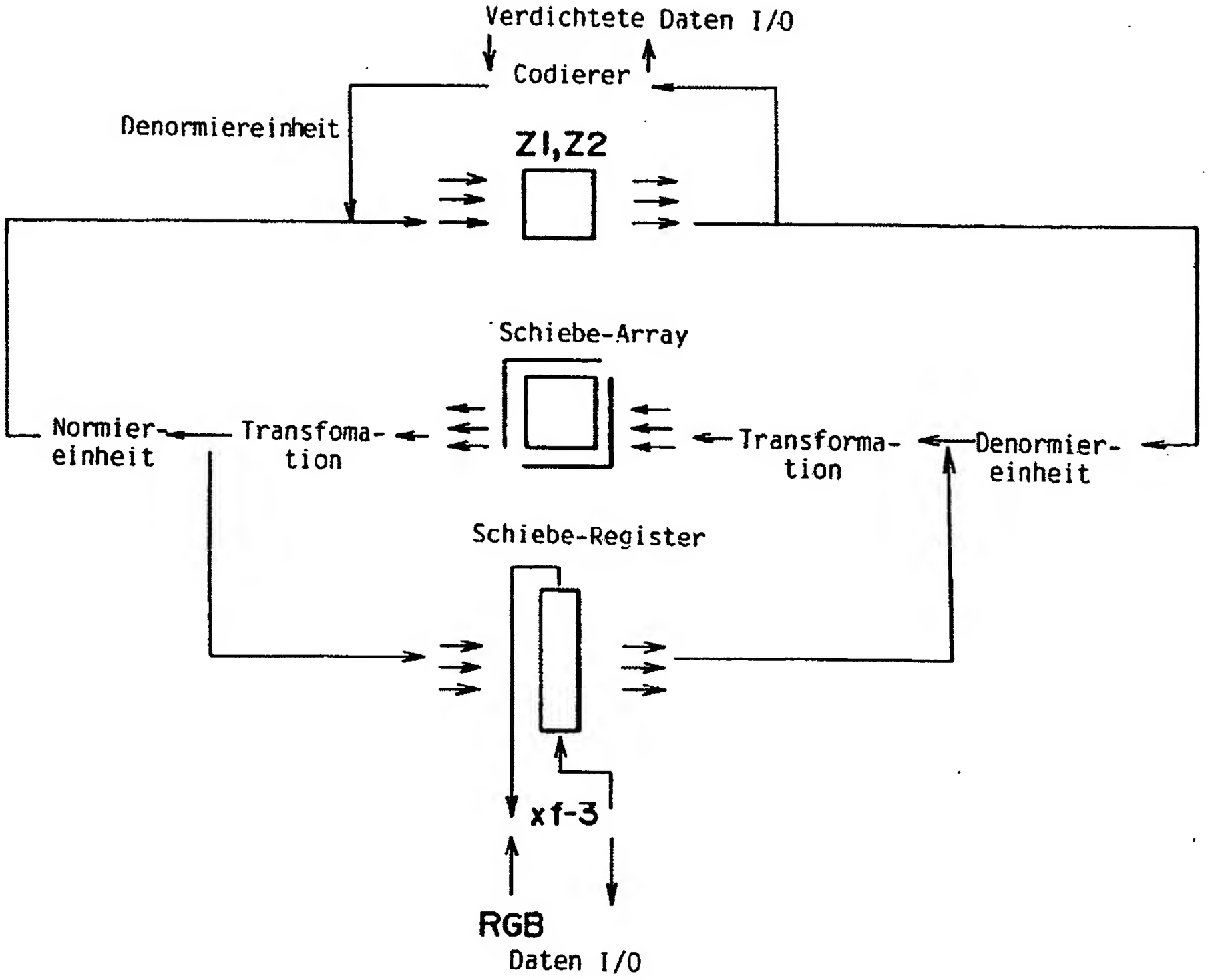
Nummer: DE 41 33 460 A1
Int. Cl.[5]: G 06 F 15/66
Offenlegungstag: 15. April 1993

FIG. 6B



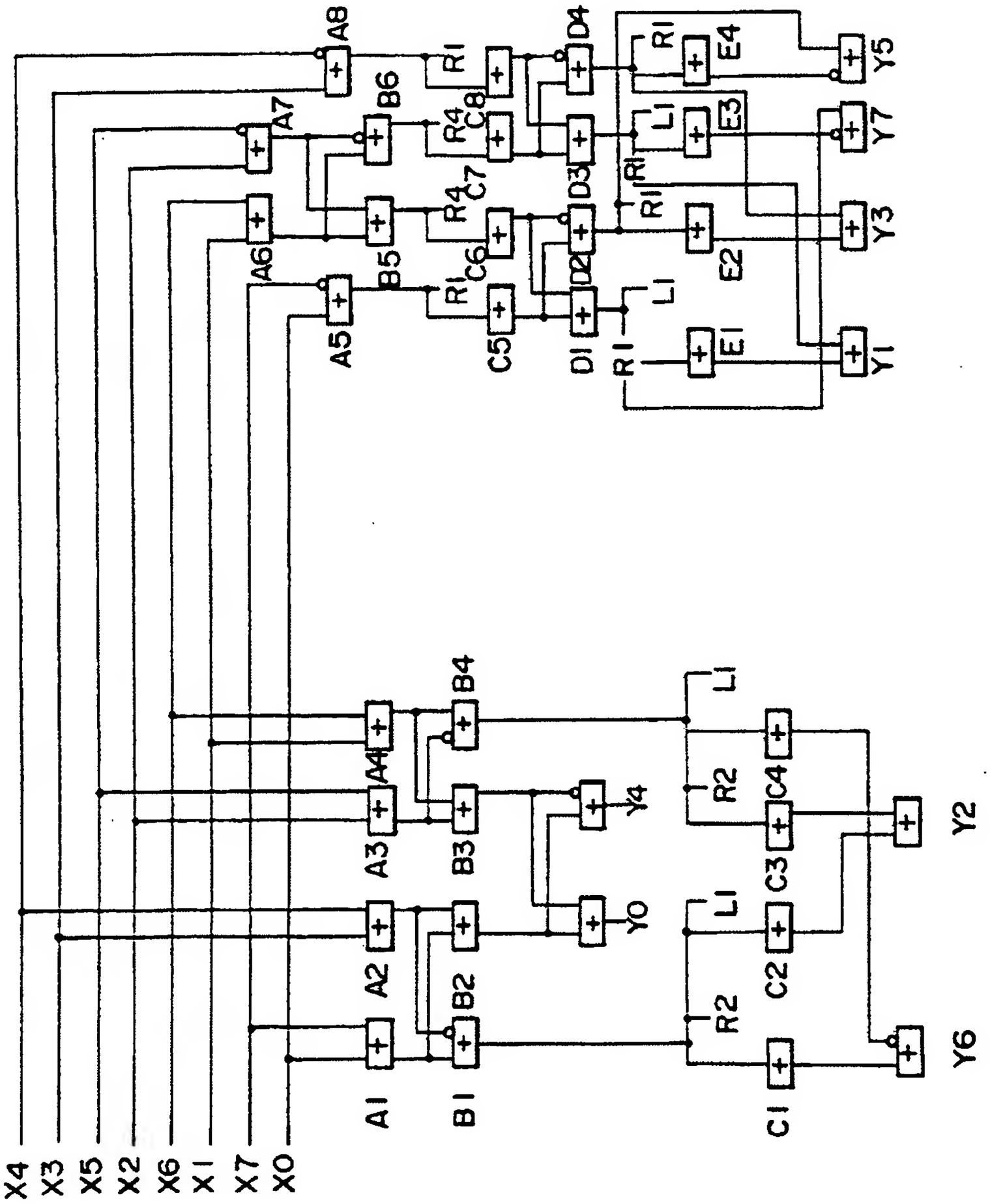
Nummer: DE 41 33 460 A1
Int. Cl.[5]: G 06 F 15/66
Offenlegungstag: 15. April 1993

FIG. 7



Nummer: DE 41 33 460 A1
Int. Cl.[5]: G 06 F 15/66
Offenlegungstag: 15. April 1993

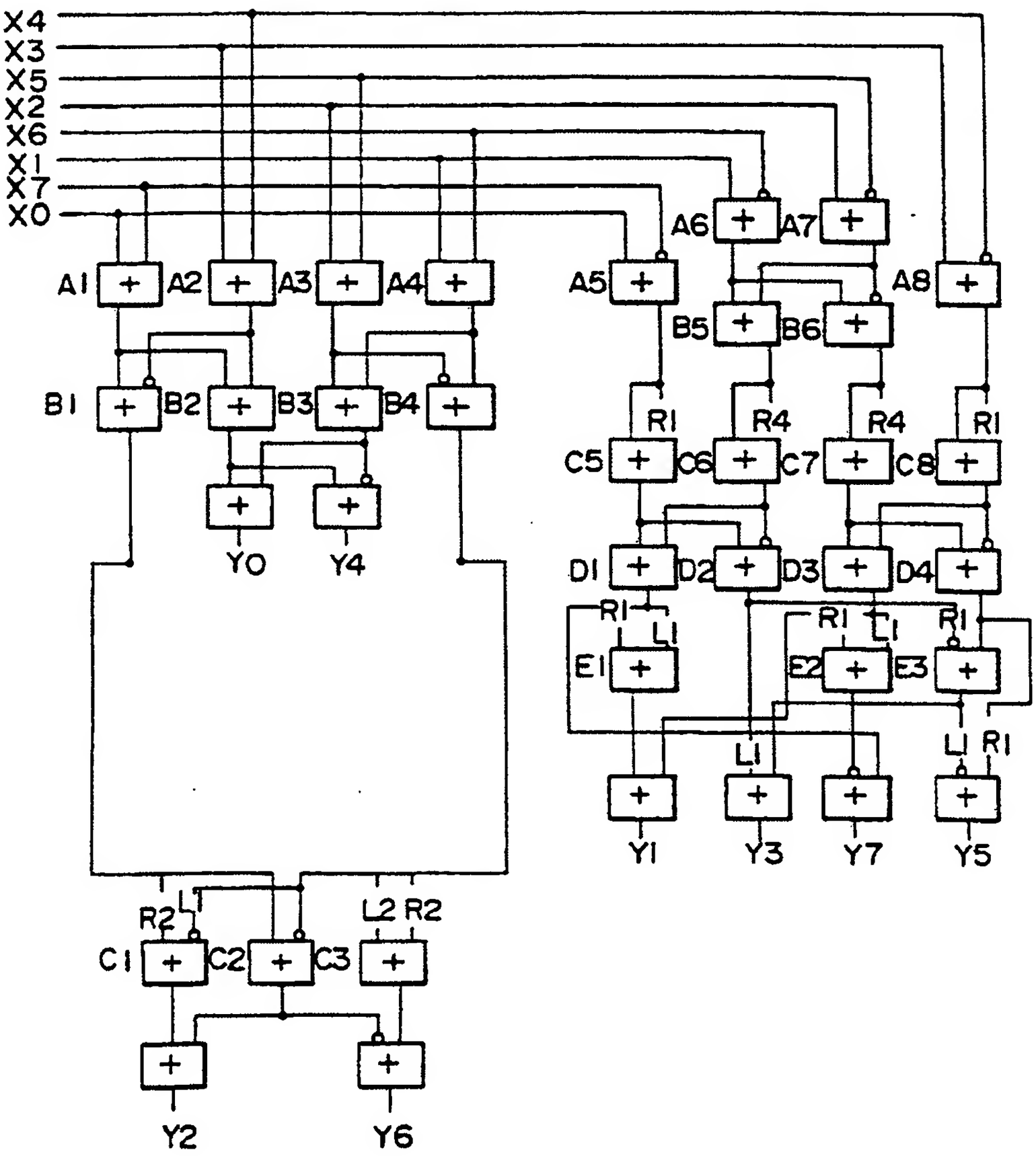
FIG. 8A



308 015/128

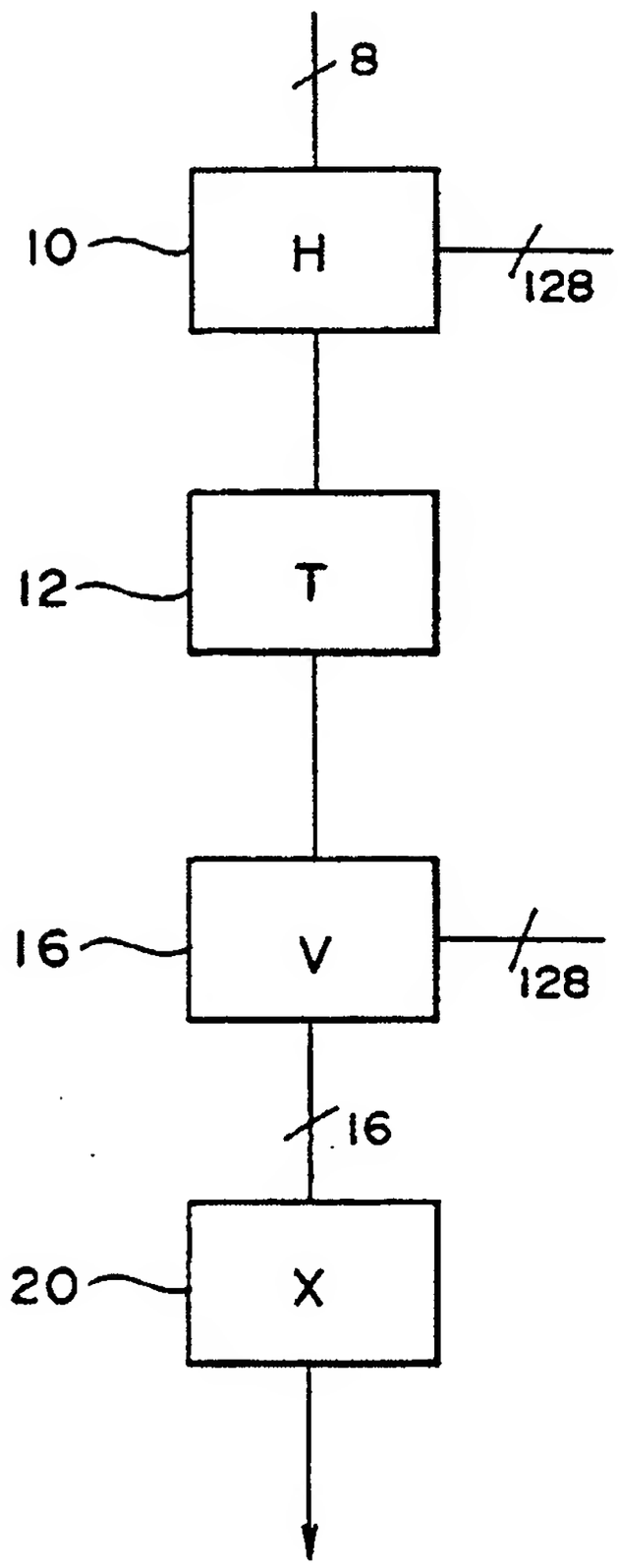
Nummer: DE 41 33 460 A1
Int. Cl.[5]: G 06 F 15/66
Offenlegungstag: 15. April 1993

FIG. 8B



Nummer: DE 41 33 460 A1
Int. Cl.[5]: G 06 F 15/66
Offenlegungstag: 15. April 1993

FIG. 9



Nummer: DE 41 33 460 A1
Int. Cl.[5]: G 06 F 15/66
Offenlegungstag: 15. April 1993

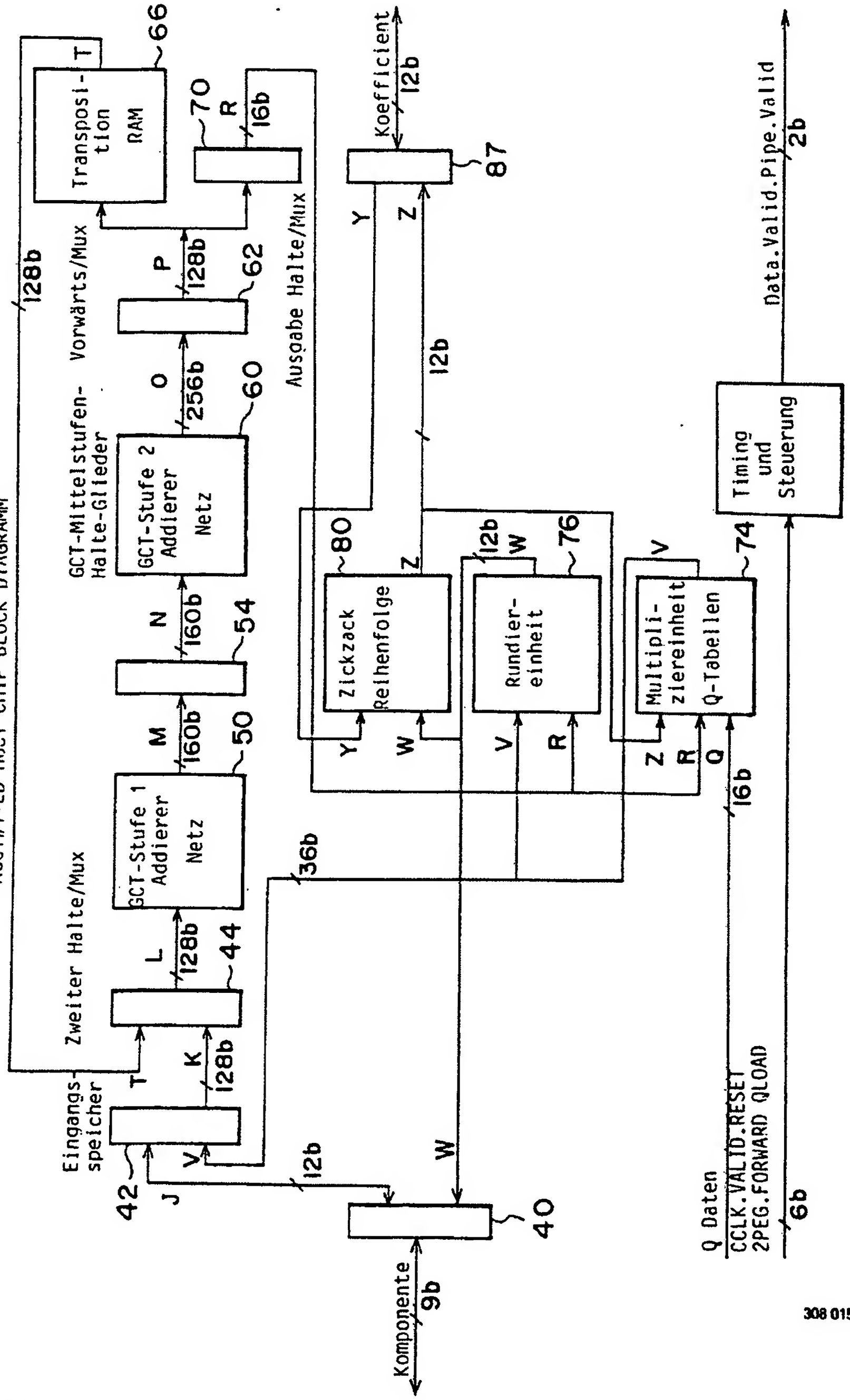


FIG. 10

308 015/128